Advanced Collection Configurations

A comprehensive set of online and offline tools are available to discover and retrieve information from an operational network for input into a plan file. There are various methods available, depending on sources of information and network access, such as SNMP access and router configuration files, and what information is to be imported.

Advanced collection configuration includes the following topics:

- Multi-Network Collection—Describes how to collect data from multiple networks and insert them into either an external archive or directly into the WAE Live data store.
- Offline Discovery—Describes the tools used to discover and retrieve information from router configurations and from RRD tools.
- Network Access File—Describes how to customize network access files that store network access parameters, such as time-out and retry settings.
- Network Authentication—Describes how to configure the authentication file. The file keeps SNMP community and router login authentication information for use by WAE Collector.
- Manage Archives—Describes the basic archive tools that apply to both the WAE Live and WAE Design Archive applications when using an augmented or manual discovery method.

Terminology

This chapter uses the following terms.

- `$WAE_ROOT`—Location of the installation. By default, this is `/opt/cariden`. These terms are interchangeable.
- `$WAE_HOME`—Directory in which the WAE Design, WAE Live, and WAE Collector executables and binaries are installed. The default is `/opt/cariden/software/mate/current`.

Multi-Network Collection

Note This section describes running multi-network collections using the manual collection method. You can also use the WAE UI to run multi-network collections. For more information on using the WAE UI, see Add Additional Networks for Collection.

The manner in which you configure multiple networks for inclusion in WAE Live depends on whether you are using an external archive or directly inserting plan files into WAE Live. This chapter describes both methods.

- Prerequisites
- Pre-Snapshot Configuration
- Using External Archives
- Inserting Plan Files Directly

This chapter references the following terms.

- `$CARIDEN_ROOT`—Location of the installation. By default, this is `/opt/cariden`.
- `$CARIDEN_HOME`—Directory in which the WAE Design, WAE Live, and WAE Collector executables and binaries are installed. The default is `/opt/cariden/software/mate/current`.

Note All instructions and examples assume you used `/opt/cariden` as the default installation directory. If you did not, then substitute your installation directory for `/opt/cariden`.

Prerequisites

This chapter does not describe the details of configuring snapshots. Rather, **it describes only the nuances of configuring manual snapshots for the purpose of discovering multiple networks**. Using this chapter has several "knowledge" prerequisites, as follows.

- How to configure both snapshot .txt and .inc files, and their relationships.
- How to configure manual snapshots, including steps not covered here, such as configuring snapshots to discover and model the network.
- Plan file insertion tools (`archive_insert` and `ml_insert_plan`). For information, refer to their `-help` output.

For information on configuring snapshot files and configuring manual snapshots, see Snapshot Files.

Best practice: Back up all configuration files before you begin.

Pre-Snapshot Configuration

Note This chapter uses two running examples. One is the collection for an "east" network where plan files are put into an external archive. The other is the collection for a "north" network where plan files are directly inserted in WAE Live. Such names are for example purposes only.

Note You can use the same authentication and network access files for all networks, or you can create and modify them on a per-network basis. For more information on these files, see Network Access File and Network Authentication.

Step 1 Run `mate_auth_init` once to create an authentication file (`auth.enc`) used by SNMP and login tools.

```
mate_auth_init
```

This is an interactive tool that first prompts you to choose the SNMP version and the relevant parameters. To create a different network authentication file for a network, use the `-auth-file` option. The recommendation is to use one of the default configuration paths: `~/.cariden/etc`, `$CARIDEN_HOME/etc`, or `$CARIDEN_ROOT/etc`.

Step 2 Optional: Customize network access. To create a different network access file, copy the default `$CARIDEN_HOME/etc/net_access.txt`, rename, and modify it. This file must be located in one of the default configuration paths: `~/.cariden/etc`, `$CARIDEN_HOME/etc`, or `$CARIDEN_ROOT/etc`.

Step 3 For new installations, copy the default snapshot.txt and snapshot.inc files to working configuration files. Uniquely name each set of .txt and .inc files to represent the network to which it is applicable.

Examples:

```
cp /opt/cariden/software/mate/current/etc/snapshot.txt /opt/cariden/etc/ss-east.txt
cp /opt/cariden/software/mate/current/etc/snapshot.inc /opt/cariden/etc/ss-east.inc
cp /opt/cariden/software/mate/current/etc/snapshot.txt /opt/cariden/etc/ss-north.txt
cp /opt/cariden/software/mate/current/etc/snapshot.inc /opt/cariden/etc/ss-north.inc
```

If this is not a new installation, you can use existing snapshot files in `/opt/cariden/etc` and make modifications noted in this chapter as needed. However, you need one set of snapshot files (one .txt file and one .inc file) per network.

Using External Archives

Each network must have its own set of snapshot files that independently call `archive_insert` to insert plan files into a uniquely named archive. Remember, you must also configure the snapshots to discover, poll, and build the network model.

Step 1 Edit the ss-east.txt file, which contains the collection, polling, modeling, and insertion tasks to perform. This file controls the sequence of execution and also contains environment variables of common values used in the ss-east.inc file.

a. At minimum, you must define `unique`, `seed_router`, `igp`, `home_dir`, and `archive_dir`. By default, the `archive_insert` tool uses the `archive_dir` environment variables when inserting plan files into an external archive. Best practice is to use the default.

Example:

```
unique east
seed_router 10.10.10.11
igp ospf
ospf_area 0.0.0
home_dir /opt/cariden
work_dir $(home_dir)/work
archive_dir $(home_dir)/archives
```

b. Edit the `include` environment variable to read the ss-east.inc file from `$(home_dir)/etc`.

Example: `include $(home_dir)/etc/ss-east.inc`

c. Uncomment the ARCHIVE_INSERT task.

Step 2 As needed, edit the ss-east.inc file to modify and add tools that are to be called from the ss-east.txt file.

Configure the file to use `archive_insert` to insert plan files into the named external archive directory. You can use the default `archive_insert` configuration in the .inc file for this purpose.

Step 3 Run `archive_init` to initialize the archive repository into which the plan files are to be inserted. Text in <angle brackets> refers to environment variables that you set in the ss-east.txt file. The path entered for the External Archive on the WAE Live Settings > General Settings page must match this case-sensitive path.

```
archive_init -archive /opt/cariden/archives/<unique>-archive
```

Example: `archive_init -archive /opt/cariden/archives/east-archive`

Step 4 Create a cron job that repeats the process of running the snapshot files that you created, which results in the insertion of plan files going into their respective archive repositories.

Note Both CARIDEN_ROOT and CARIDEN_HOME variables must be defined from within the crontab. You cannot use CARIDEN_HOME=$CARIDEN_ROOT/software/mate/current.

a. Open the file for editing as follows.

```
crontab -e
```

b. At the end of the file, add the following lines.

```
CARIDEN_ROOT=/opt/cariden
CARIDEN_HOME=/opt/cariden/software/mate/current
```

c. At the end of the file, add one entry to call the unique snapshot per network.

Example:

```
*/30 * * * * $CARIDEN_HOME/bin/snapshot -config-file $CARIDEN_ROOT/etc/ss-east.txt
2>&1
```

Note Best practice is to verify the snapshots are working prior to continuing to WAE Live configurations.

Inserting Plan Files Directly

Each network must have its own set of snapshot files that independently call `ml_insert_plan` to insert data directly into the WAE Live data store. These files must also call `archive_insert` to insert plan files directly into the Map archive. Remember, you must configure the snapshots to discover, poll, and build the network model.

Step 1 Edit the ss-north.txt file, which contains the collection, polling, modeling, and insertion tasks to perform. This file controls the sequence of execution and also contains environment variables of common values used in the ss-north.inc file.

a. At minimum, you must define `unique`, `seed_router`, `igp`, `and home_dir`.

If using the Map component, create an environment variable that uniquely specifies the Map archive.

Example:

```
unique north
seed_router 10.10.10.11
igp isis
isis_level 2
home_dir /opt/cariden
```

```
work_dir $(home_dir)/work
map_archive_dir $(home_dir)/data/mldata/archive
```

b. Edit the `include` environment variable to read the ss-north.inc file from `$(home_dir)/etc`.

Example: `include $(home_dir)/etc/ss-north.inc`

c. Uncomment the ML_INSERT task.

d. If using the Map component, add an MAP_ARCHIVE_INSERT task.

Example:

```
<ARCHIVE_INSERT_TASKS>
#ARCHIVE_INSERT
ML_INSERT
MAP_ARCHIVE_INSERT
```

Step 2 As needed, edit the ss-north.inc file to modify and add tools that are to be called from the ss-north.txt file.

Configure the associated ss-north.inc file to use `ml_insert_plan` with the `-network` option. The `-network` option specifies the network partition of the data store into which you are placing data. This must match the case-sensitive network name added through the WAE Live UI. In this example, you would also have to create and name a WAE Live network called “north.”

Example:

<ML_INSERT>	
Name	Value
cmd	$(cariden_home)/bin/ml_insert_plan
cmd_opt	ML_INSERT_CMD_OPT
<ML_INSERT_CMD_OPT>	
Name	Value
plan-file	$(work_dir)/$(unique).pln
time	$(start_time_direct)
log-file	$(log_dir)/$(unique)-log-ml_insert.log
network	north

Step 3 If using the Map component, configure the associated ss-north.inc file to use `archive_insert` to specify the Map archive. This must match the case-sensitive name given the Map archive in the WAE Live Settings > General Settings page. In this example, the Map archive in WAE Live would have to be `/opt/cariden/data/mldata/archive/north`.

Example:

<MAP_ARCHIVE_INSERT>	
Name	Value
cmd	$(cariden_home)/bin/archive_insert
cmd_opt	MAP_ARCHIVE_INSERT_CMD_OPT

<MAP_ARCHIVE_INSERT>	
<MAP_ARCHIVE_INSERT_CMD_OPT>	
Name	Value
plan-file	$(work_dir)/$(unique).pln
archive	$(map_archive_dir)/$(unique)
time	$(start_time)
log-file	$(log_dir)/$(unique)-log-map-archive_insert.log

Step 4 Create a cron job that repeats the process of running the snapshot files that you created, which results in the insertion of plan files going into their respective network segments within the WAE Live data store.

Note Both CARIDEN_ROOT and CARIDEN_HOME variables must be defined from within the crontab. You cannot use CARIDEN_HOME=$CARIDEN_ROOT/software/mate/current.

a. Open the file for editing as follows.

```
crontab -e
```

b. Add the following lines.

```
CARIDEN_ROOT=/opt/cariden
CARIDEN_HOME=/opt/cariden/software/mate/current
```

c. At the end of the file, add one entry to call the unique snapshot per network.

Example:

```
*/30 * * * * $CARIDEN_HOME/bin/snapshot -config-file $CARIDEN_ROOT/etc/ss-north.txt
2>&1
```

Best practice is to verify the snapshots are working prior to continuing to WAE Live configurations.

Offline Discovery

Note Configuring `get_configs` is supported in both augmentation and manual collection methods. Using `parse_configs` to augment a plan with RSVP LSP and/or SRLG data is supported through the augmentation method. Configuring `parse_configs` to create a plan file for overall topology is supported only in manual collection.

This chapter describes the CLI tools available to discover and retrieve information from router configuration tools and from RRD tools, such as Cricket, Cacti, and MRTG.

Import Databases

The following tools are useful for capturing and importing network information. For instance, you can capture the configuration files or IGP databases and import them into WAE Collector.

- `get_configs`—Reads the configuration files from a list of routers and saves them in the specified directory.
- `parse_configs`—Reads a set of Cisco and/or Juniper Networks router configuration files, and creates a plan file of the network. See Import Router Configuration Files. For information on using this tool from the WAE Design GUI, see the *Cisco WAE Design Integration and Development Guide*.
- `parse_igp`—Converts IGP information from router `show` commands to a plan file. See Import IGP Database. For information on using this tool from the WAE Design GUI, see the *Cisco WAE Design Integration and Development Guide*.
- `get_show` and `get_xml`—Tools for entering router `show` commands and the XML equivalents for further processing by the user or an application. These commands are typically used because the WAE Collector does not include the output of the commands during plan file creation. The `get_xml` tool offers similar functions to `get_show`. It is used to get structured data from devices by executing XML commands on them. The command format is device-dependent.

Note This section contains examples for Cisco and Juniper routers. For information about network discovery of routers for other vendors, please contact your support representative.

Import Router Configuration Files

The `parse_configs` tool reads Cisco, Juniper, and Huawei router configuration files, and creates a plan file.

The router configuration files from the network, or part of the network, need to be available in a specific directory. The `parse_configs` tool reads files in this directory (`-data-dir` option), determines the router type/vendor, and parses the configuration.

The following information can be read from a router configuration file to create the plan file. After parsing this information, the tool matches corresponding interfaces in the IGP mesh to create the network topology.

• Router name • Vendor • Model • OS • Router IP address (loopback) • Management interface IP address (Cisco IOS XR and Juniper Junos) • Interface names (inside IGP topology) • Interface IP addresses • Interface capacities (if available)	• IGP type and metrics (IS-IS or OSPF) – Process ID (OSPF) – Instance ID (IS-IS) • RSVP reservable bandwidth (MPLS) • MPLS LSPs (including bandwidth and FRR LSPs) • LAG[1] ports and bundle ports • SRLGs, including which SRLGs are configured on which nodes (Cisco and Juniper) • VPN interfaces • VPN PE membership

1. LAG is specific to Ethernet. Bundle is generic and applies to different link types.

With the `-igp-protocol` option, you can select which interfaces are part of the topology: IS-IS and/or OSPF enabled interfaces. The default is `isis`.

- For IS-IS networks, the tool can read IS-IS Level 1, Level 2, or both Level 1 and Level 2 metrics. If both are selected, `parse_configs` combines both levels into a single network, and Level 2 metrics take precedence. The `-isis-level` option specifies which option to use; the default is Level 2.
- For OSPF networks, the tool can read information for single or multiple areas. The `-ospf-area` option specifies the area ID or all. The default is area 0.

ASN is ignored by default. However, for networks that span multiple BGP ASNs, use the `-asn` option to read information from more than one IGP process ID or instance ID in an ASN.

Shared media segments in the network (non point-to-point circuits, such as Ethernet) are included in the topology by default unless the `-shared-media` option is set to `false`. A pseudonode and interface representing the medium are then created for every shared medium with more than two hosts, as used by OSPF and IS-IS routing protocols.

With the `-plan-file` option, you can merge an existing plan file with router configurations to create an augmented plan file. For example, you could use the `parse_igp` output as the input into `parse_configs`.

Note A useful tool for maintaining an archive of router configuration files is RANCID (`http://www.shrubbery.net/rancid/`).

Import IGP Database

The parse_igp tool reads one or more databases that are generated from a router's CLI. With the `-igp-protocol` option, you can select an IGP protocol.

- IS-IS IPv4 or IPv6 using the `isis` or `isisv6` option, respectively
- OSPF IPv4 or IPv6 using the `ospf` or `ospfv3` option, respectively

With the `-plan-file` option, you can merge an existing plan file with the IGP databases to create an augmented plan file. For example, you could use the `parse_configs` output as the input into `parse_igp`.

IS-IS

This tool can generate a topology out of an IS-IS Level 1, Level 2, or both databases using the `-level` option.

To capture an IS-IS database from the CLI, log into a router within the IS-IS topology, display the IS-IS database, and save the output of that session to a file, as follows.

Step 1 Establish a terminal session on a host that has direct access to the network routers, for example, using telnet or SSH.

Step 2 Initiate a process to capture the entire session.

Step 3 Log on to the seed router, which is a router that contains IGP information for the network.

Step 4 Disable paging of output by setting the terminal length to infinite (0).

Cisco:	`terminal length 0`
Juniper:	`set cli screen-length 0`

Step 5 Cisco Option: Disable dynamic host resolution in IS-IS if hostnames are longer than 14 characters and the unique part of the name is after 14 characters. Cisco routers truncate names at 14 characters. To disable dynamic host resolution, enter the `router isis <proc id>` command mode, and then enter:

```
no hostname dynamic
```

Step 6 Display the IS-IS link-state database (LSDB).

Cisco:	`show isis database verbose`
Juniper:	`show isis database extensive`

Step 7 Log out of the router, the network host, and the screen capture, each time using the `exit` command.

Step 8 Save your session capture that was initiated in step #2 (`exit` when using `script`).

Repeat the above steps to capture IS-IS databases from additional routers (Level 1 and Level 2), if necessary. The resulting file or directory includes login and logout commands, as well as output. Now you can use the `parse_igp` tool.

- Use the `-level` option to specify whether discovering Level 1 or Level 2 topology or both; the default is level 2.
- Pass the file created in the above steps using the `-database-file` option.
- If there are multiple files (multi-level topologies), pass the directory name where the files are located using the `-database-dir` option.

Example: This command uses IS-IS Level 2 topology information stored in the `mobile_database.txt` file to create a plan file called `mobile_model.txt`.

```
parse_igp -igp-protocol isis -database-file mobile_database.txt -out-file mobile_model.txt
```

IS-IS Database Information

By default, the IS-IS protocol used in IP networks (non MPLS) does not distribute the IP addresses of the interfaces in the network, nor the circuit capacities.

When the IS-IS TE-extensions (for MPLS) have been enabled in the network, that information becomes available, and will also be used by `parse_igp`.

- Cisco—You must specifically enable the TE extensions using `mpls traffic-eng level-2` in the `router isis` configuration section, and `mpls traffic-eng tunnels` on the interfaces. Doing so makes both the IP addresses and circuit capacities available in IS-IS (and `parse_igp`).
- Juniper Networks—IS-IS TE extensions are enabled by default, and IP addresses are available for all interfaces in those routers. If RSVP is also enabled on an interface, the capacity of that circuit is available in IS-IS.

Enable the `-use-dns` option by setting it to `true` if DNS (domain name server) needs to resolve IP addresses (router names) in the IS-IS database file.

Note Parallel circuits (non-TE enabled) between two Cisco routers, show up in the IS-IS database as a single circuit.

OSPF

To capture an OSPF database from the CLI, log into a router with the OSPF topology, display the OSPF database, and save the output of that session to a file.

Step 1 Establish a terminal session on a host that has direct access to the network routers, for example, using telnet or SSH.

Step 2 Initiate a process to capture the entire session.

Step 3 Log on to the seed router. This is a router that contains IGP information for the network.

Step 4 Disable paging of output by setting the terminal length to infinite (0).

Cisco: `terminal length 0`

Juniper Networks: `set cli screen-length 0`

Step 5 Follow the appropriate Cisco or Juniper Networks step.

Cisco	Juniper Networks
If the system supports DNS, enable it so the database includes host names, rather than just IP addresses. Enter the configuration command mode, and then enter this command. IOS: `ip ospf name-lookup` IOS XR: `ospf name-lookup`	Display the OSPF database. If the system supports DNS, pipe the output of the `show` command to the resolver so the database has host names, rather than just IP addresses. `show ospf database extensive | resolve`
Display the OSPF database. IOS: `show ip ospf database router` IOS XR: `show ospf database router`	Otherwise, just show the database, which identifies routers by IP address only. `show ospf database extensive`
Display the OSPF TE database. IOS: `show ip ospf database opaque-area` IOS XR: `show ospf database opaque-area`	Display OSPFv3 database. `show ospf3 database extensive`

Display OSPFv3 database (IOS only)
`show ipv6 ospf database router`
`show ipv6 ospf database link`
`show ipv6 ospf database prefix`	

Step 6 Log out of the router, the network host, and the screen capture, each time using the `exit` command.

Step 7 Save your session capture that was initiated in step #2 (`exit` when using `script`).

Repeat the above steps to capture OSPF databases from additional area border routers (ABRs), if necessary. The resulting file or directory includes login and logout commands, as well as output. Now you can pass the file created in the above steps to using the `parse_igp` tool using the `-database-file` option. If there are multiple files, then pass the directory name where the files are located using the `-database-dir` option.

By default, `parse_igp` collects the OSPF area 0 link-state database (LSDB). To generate topologies from non-zero area LSDBs, use the `-ospf-area all` option. The tool then identifies all ABRs and builds a complete multi-area OSPF network topology. Note that the `login_find_igp_db` tool uses this `-ospf-area all` option as well.

OSPF Database Information

Unlike the IS-IS database, the OSPF database has IP address information for all interfaces in the network. If the network is TE-enabled, the OSPF database also contains circuit capacities.

Enable the `-use-dns` option by setting it to `true` if DNS needs to resolve IP addresses (router names) in the IS-IS database file.

Note Parsing IGP with the OSPF protocol option only processes area 0 routers per default. Use the `-area` option to select another area, or `all` for all areas.

get_show and get_xml

The `get_show` tool is a wrapper for entering a `show` command on one or more routers. For example, the `get_show` tool with a `-cmd` argument of `show configuration` is equivalent to the `get_configs` tool. The `-command-table` option enables you to enter vendor-specific CLI commands, such as an ICMP ping in multi-vendor networks. You could also use this tool to get an OSPF or IS-IS database from the router.

Because `show` commands are highly dependent on router types, this tool can only operate on a homogeneous set of routers when more than one is specified. The IS-IS and OSPF `show` commands are listed in the IS-IS and OSPF sections, respectively.

In the `-nodes-table` or `-nodes` arguments, if an IP address is available, it is used. Otherwise, an IP lookup through DNS is tried. If that fails, an error is returned.

The `get_xml` tool offers the same function as `get_show`. It is used to get structured data from devices by executing XML commands on them. The command format is device-dependent.

Import Traffic from RRD Tools

WAE Collector can import network information from the following RRD tools.

- Cricket
- Cacti
- MRTG

Cricket

The `cricket_poll_interfaces` tool reads a router interfaces file, discovers which interfaces the file specifies, and the RRD files that contain the data associated with each interface. It then reads the traffic measurements from the RRD file and imports them into to the <InterfaceTraffic> and <NetIntIfMeasurements> tables in a plan file.

Cacti

Because Cacti is written in PHP and uses mysql, the importer is also implemented with a PHP script. Install the PHP script on the web server that is running Cacti, and then invoke `cacti_poll_interfaces` to import the traffic measurements into the <InterfaceTraffic> and <NetIntIfMeasurements> tables in a plan file.

```
http://.../cacti/graph_info.php?get=tab
```

To install the PHP script on a web server running Cacti, follow these steps.

Step 1 Copy the PHP script to the web server. You must have a guest account set up for Cacti. The script location is as follows.

```
$CARIDEN_HOME/lib/php/cacti/graph_info.php
```

Step 2 Add `"graph_info.php" => 7`, to `include/global_arrays.php`. The array location is as follows.

```
$user_auth_realm_filenames
```

To import traffic measurements into a plan file, call `cacti_poll_interfaces` and provide the Cacti URL as an argument.

```
http://.../cacti/graph_info.php?get=tab
```

MRTG

The `mrtg_poll_interfaces` tool imports traffic measurements into a plan file by reading an MRTG configuration file. First it discovers which interfaces the configuration file specifies, along with the RRD files that contain the data associated with each interface. Then it reads the RRD files and imports the traffic measurements into the <InterfaceTraffic> and <NetIntIfMeasurements> tables in a plan file.

Network Access File

Note Editing the network access file is supported for the WAE Collector UI collection and for the manual collection methods.

A network access file can be used to store network access parameters. These include timeout and retry settings, and settings for management of multiple simultaneous queries. Having these settings in a file, rather than as CLI parameters, removes the redundancy across many calls and allows for more complex settings (per router settings, for example).

The network access file provides default settings for all access parameters. You can use either the default network access file, or you can modify and put it in one of the following locations. The file is looked for in this sequence, and the first version found is used.

- `~/.cariden/etc`
- `$CARIDEN_ROOT/etc`
- `$CARIDEN_HOME/etc` (default)

When the Collector server uses this file, it saves it as net_access_session.txt file. Augmented snapshots then use the net_access_session.txt file that was us chapter.

Best Practices

- Make a copy of the default net_access.txt file located in `$CARIDEN_HOME/etc` before editing it, and ed in the last collection by the Collector server. For information on configuring the Collector server, see Collecting Basic Information Using the WAE Collector UI and put the edited version in `$CARIDEN_ROOT/etc`. This simplifies the upgrade process and preserves a copy of the original if needed.
- When upgrading, compare the net_access.txt file in the new release to the one in the existing release to determine if your edits need to be incorporated into the new net_access.txt file.

File Format

The network access file consists of two sections: one containing tables that set values globally and one containing tables that sets values on a per-router basis.

Note In the net_access.txt file an empty field means *everything else*, and this meaning is in context of the rows defined before it. If it is in the first row, it means *everything*.

Global Settings

WAE Collector network communication tools take advantage of the polling abilities that simultaneously process a large number of network requests. The Global section of the network access file defines constraints that are used to limit the impact to either the server doing the polling or to the network elements between the server and the network being polled. Examples of network elements that could be heavily impacted by polling traffic are a firewall, slow WAN circuits, or a NAT device.

This section consists of two tables that work in tandem: <GlobalModes> and <GlobalSettings>.

- <GlobalModes>—This table groups together settings that are used to constrain the speed of the network communications. These settings are grouped into names (in the Name column), and are activated by referencing them in the GlobalMode column of the <GlobalSettings> table. These names are user-definable.

  The network access file includes commented documentation for each <GlobalModes> property. Table 4-1 provides an example <GlobalModes> table.

- <GlobalSettings>—This table defines the association between the entries in its TaskRegExp column and the entries in the <GlobalModes> Name column.
  - TaskRegExp—This is the WAE Collector CLI tool. The default is a blank, which matches all possible tools.
  - GlobalMode—Mode to assign to all routers when running the matched CLI tool.

Table 4-2 provides an example <GlobalSettings> table. The empty field at the beginning of the last row means *everything except* `snmp_poll` and `snmp_find_*`.

Table 4-1 Example <GlobalModes> Entries

Name	Property	Value
Normal	SNMP_max_queries_total	1000
Normal	SNMP_max_open_session	200
Normal	SNMP_collection_interval	120000
Normal	LOGIN_max_open_sessions	10
Normal	LOGIN_session_open_interval	0
Slow	SNMP_max_queries_total	500
Slow	SNMP_max_open_session	50
Slow	SNMP_collection_interval	240000
Slow	LOGIN_max_open_sessions	2
Slow	LOGIN_session_open_interval	1
Fast	SNMP_max_queries_total	2000
Fast	SNMP_max_open_session	400
Fast	SNMP_collection_interval	60000
Fast	LOGIN_max_open_sessions	20
Fast	LOGIN_session_open_interval	0

Table 4-2 Example <GlobalSettings> Entries

TaskRegExp	GlobalMode
snmp_poll	Fast
snmp_find_*	Slow
	Normal

Per Router Settings

If you have concerns about specific device types or operating systems, you can constrain the WAE Collector network communication tool to execute on a per-router basis. For example, some devices might not respond well to short SNMP timeout values when they are busy, while others might need special settings for login access. Together, the <RouterModes> and <PerRouterSettings> tables enable you to adjust these types of settings.

- <RouterModes>—This table defines groups of devices to either block or constrain their communications. For each name (in the Name column) that you create, you must enter a value for all SNMP properties.

  The network access file includes commented documentation for each <RouterModes> property. Table 4-3 provides an example <RouterModes> table.

- <PerRouterSettings>—This table associates named groups of <RouterModes> parameters with a specific set of devices within the network. Each Name entry in the <RouterModes> table has a corresponding entry in the RouterMode column.

  Each RouterMode is defined by the NodeRegExp, IPRegExp, and SQLFilter columns.

  - NodeRegExp is matched against device names.
  - IPRegExp is matched against device IP addresses.
  - SQLFilter is an SQLite `sql` command that can reference any column of the Nodes table to match devices.

  The TaskRegExp column provides constraints for one specific tool in the event that unique parameters are required for one discovery task.

  Table 4-4 provides an example <PerRouterSettings> table. The empty fields in the first row mean *everything*. The empty TaskRegExp field in the last row means *everything except* `snmp_find_multicast` and `snmp_poll`.

Table 4-3 Example <RouterModes> Entries

Name	Property	Value
Normal	SNMP_max_timeout	3
Ignore	SNMP_max_timeout	0
Limit_CRS	SNMP_max_timeout	3
Multicast_Login	SNMP_max_timeout	3
Multicast_SNMP	SNMP_max_timeout	3
Junos_old	SNMP_max_timeout	3
Junos_new	SNMP_max_timeout	3
Normal	SNMP_RSVP_stats_method	Default
Junos_new	SNMP_RSVP_stats_method	Method1
Junos_old	SNMP_RSVP_stats_method	Method2

Table 4-4 Example <PerRouterSettings> Entries

NodeRegExp	IPRegExp	TaskRegExp	RouterMode	SQLFilter
			Ignore	Name REGEXP '^sl-gw.*'
		snmp_find_multicast	Ignore	Name NOT REGEXP 'sl-crs.*' AND Name NOT REGEXP 'sl-bb.*'
		snmp_poll	Limit_CRS	OS REGEXP '^IOS XR.*'
			Normal	

Discovering Multi-Vendor Networks with SAM

If you are discovering a network containing both Alcatel and non-Alcatel nodes, you must configure the <PerRouterSettings> table to tell the online tools to ignore the Alcatel objects and their traffic. The simplest method is to do the following.

Step 1 Add a comment (#) to this line to prevent the collection of Alcatel statistics.

```
# snmp_pollALU_REALTIMEOS REGEXP '^TiMOS.*'
```

Step 2 Uncomment this line to ignore the discovery of Alcatel nodes, interfaces, and LSPs, and to ignore the collection of statistics from them.

```
(snmp_find_nodes|snmp_find_interfaces|snmp_find_rsvp|snmp_poll)IgnoreVendor =
'Alcatel-Lucent'
```

Test the Network Access File

The `mate_access_test` tool enables you to specify a node, node IP, and task, or alternatively specify the router mode and global mode settings directly. The tool returns the global and per-router parameter settings that are applied if the network access file were used. The option is `-net-access-file`. The default value is net_access.txt.

Use `mate_access_test -net-access-file` to see the global and per-router parameter settings that are applied if a network access file is specified. The default value for `-net-access-file` option refers to the net_access.txt file in the configuration path.

Tool Access Parameters

Each WAE Collector online tool (for example, `snmp_find_interfaces`) contains three parameters to control network access settings.

- `-net_access_file <file>`—Overrides the default network access file.
- `-net_access-router-mode <name>`—This name specifies the RouterMode that overrides the <PerRouterSettings> table.
- `-net-access-global-mode <name>`—This name specifies the GlobalMode that overrides the <GlobalSettings> table.

Network Authentication

Note Creating and editing the authentication file is supported for the manual collection method.

The authentication file consolidates the login, authentication, encryption, community strings, and other credentials needed by the WAE Collector tools to access routers and collect network data. It is required if the tools are to be called by scripts, or if different routers in the network require different authentication information. The file can be encrypted for security and protected with a master password.

- Manual snapshots—Use the auth.enc authentication file. The `mate_auth_init` tool simplifies the process of creating a default authentication file.
- Augmented snapshots—Use the auth_session.enc file that was used in the last collection by the Collector server. The password for de-encrypting this file is set in the Node Access page of the WAE Collector UI.

Online Discovery Authentication

When an online discovery tool needs authentication information for a router (for example, `snmp_find_interfaces` needs a community string to perform an SNMPv2c query), it accesses the authentication file and looks for a match for the router. If successful, the tool uses the credentials from the file to access routers and collect network data. Without a match the tool generates a prompt or notification.

- SNMPv2c—Prompts for authentication credentials and proceeds.
- SNMPv3—Notifies the user to create an authentication file and terminates.

You can disable user interaction by setting the `-auth-prompt` option to `false`.

Create an Authentication File

Note Use this method of creating a network authentication file only if using the manual snapshot collection process.

The `mate_auth_init` tool is an interactive tool that simplifies the process of specifying a default set of authentication credentials that WAE Collector tools use to access all routers. The file is created in the directory from which you execute the command. To change the file location, enter a full path name.

The file it creates has credentials for SNMPv2c, SNMPv3, or both. SNMPv2c uses a less secure security model, passing community strings in clear text. SNMPv3 provides a strong security model that supports authentication, integrity, and confidentiality.

If `mate_auth_init` does not find an auth.enc file in one of the default locations, the tool prompts you to select one from a list.

- `~/.cariden/etc`
- `$CARIDEN_HOME/etc`
- `$CARIDEN_ROOT/etc` (Linux only)

The tool creates a file named `auth.enc` in the selected directory. However, you can override the default directory and filename by using the `-auth-file` option. The recommendation is to use one of the above default configuration paths. If you put this file in a different directory, binaries must be explicitly called using this path.

Example: `mate_auth_init -auth-file /opt/cariden/etc/auth-acme.enc`

The `mate_auth_init` tool prompts you to choose SNMPv2c, SNMPv3, or both. Depending on your choice, the tool prompts you for authentication information that is pertinent to the selected SNMP version.

Note If both SNMPv2c and SNMPv3 are selected, the default is for the auth.enc file to put all nodes in both SNMPv2c and SNMPv3. When a node is mapped to both, then only SNMPv3 is used. To change this behavior, decrypt the auth.enc file using `mate_auth_export`, edit the authentication tables based on the IPRegExp values, and then re-import the file using `mate_auth_import`.

The authorization file password and default seed router login credentials consist of the following.

- master password—Password for viewing file contents
- login username—Default username for login access to the routers

- login password—Default password for login access to the routers
- login enable password—Default enable password for login access

The SNMPv2c information is defined using a single value.

- community—Default community string

The SNMPv3 information defines authentication and encryption details.

- Security level
  - noAuthNoPriv—Authenticates by username, but does not validate the user and does not encrypt data.
  - authNoPriv—Authenticates by username, validates the user using MD5 or SHA, but does not encrypt data.
  - authPriv—Authenticates by username, validates the user using MD5 or SHA, and encrypts data using DES or AES.
- SNMPv3 username—Username for authentication
- Authentication protocol type—MD5 or SHA
- Authentication password—Password for authentication
- Encryption protocol type—DES or AES
- Encryption password—Password for encryption
- Context name—Name of a collection of management information accessible by an SNMP entity

After you have created the initial encrypted authentication file, you can manually edit the contents to add multiple profiles or communities and map routers to them. Each profile contains a complete set of SNMPv3 authentication and encryption information. Multiple profiles or communities are necessary when different groups of routers use different authentication credentials. For information about editing an encrypted authentication file, see Add Router-Specific Authentication Information.

Tables in the Authentication File

The contents of the encryption file are organized into tables.

- <MasterPassword>—Contains the master password for viewing or changing the file (Table 4-5).
- <UserTable>—Contains usernames and passwords for login access to nodes (Table 4-6).
- <CommunityTable>—Contains SNMPv2c community strings for access to nodes (Table 4-7).
- <SNMPv3ProfileTable>—Contains SNMPv3 profiles, which define a set of authentication, encryption, and context information (Table 4-8).
- <SNMPv3MappingTable>—Defines how to match routers with the SNMPv3 profiles (Table 4-9).

Note If both SNMPv2c and SNMPv3 are selected, the default is for the auth.enc file to put all nodes in both SNMPv2c and SNMPv3. When a node is mapped to both, then only SNMPv3 is used. To change this behavior, decrypt the auth.enc file using `mate_auth_export`, edit the authentication tables based on the IPRegExp values, and then re-import the file using `mate_auth_import`.

Table 4-5 <MasterPassword> Format

Column	Description
Password	Optional: Password for accessing the authentication file. If table or password is missing, the authentication file is unencrypted plain text and no password is required to view or change the file contents.

Table 4-6 <UserTable> Format

Column	Description
IPRegExp	Regular expression to match node IP addresses; if missing, defaults to accept all.
Username	Username for login access.
Password	Password for login access.
EnablePassword	Enable password for login access.

Table 4-7 <CommunityTable> Format

Column	Description
IPRegExp	Regular expression to match node IP addresses; if missing, defaults to accept all.
Community	Community string for SNMP access.

Table 4-8 <SNMPv3ProfileTable> Format

Column	Description
Profile Name	Descriptive name of the routers to which this profile applies.
Security Level	Level of SNMP security. Value is noAuthNoPriv, authNoPriv, or authPriv.
Username	User for which SNMP services are provided.
Auth Protocol	Protocol for authenticating the user. Values are MD5 or SHA. Required if using authNoPriv or authPriv security level.
Auth Password	Authentication password. Required if using authNoPriv or authPriv security level, and must be equal to or greater than eight characters.
Encryption Protocol	Protocol for encrypting data. Values are DES or AES. Required is using authPriv security level.
Encryption Password	Encryption password. Required if using authPriv security level, and must be equal to or greater than eight characters.
Context Name	Optional: A collection of management information accessible by an SNMP entity. If one or more context names are configured on a router, then a value is required. You can enter one context name only, and it is used to access all routers.

Table 4-9 <SNMPv3MappingTable> Format

Column	Description
IPRegExp	Regular expression to match node IP addresses; if missing, defaults to accept all.
Profile Name	Name of a profile in the <SNMPv3ProfileTable>.

Add Router-Specific Authentication Information

You can add additional router-specific information to the authentication file by adding rows to the authentication file tables (see Tables in the Authentication File). For router login and authentication, edit the <UserTable>. For SNMP, the following tables apply.

- SNMPv2c—Add community strings to the <CommunityTable> and map routers to these communities with a regular expression in the `IPRegExp` column.
- SNMPv3—Add security profiles to the <SNMPv3ProfileTable> and map routers to profiles in the <SNMPv3MappingTable> with a regular expression in the `IPRegExp` column.

If the authentication file is encrypted using a master password, you must first export the contents to plain text using the `mate_auth_export` tool, edit the tables using a text editor, and then encrypt it using `mate_auth_import`.

For SNMPv2c communities only, a more convenient method is provided by `auth_try_communities`. First provide a list of nodes (routers), for example from a plan file obtained through parsing the IGP database. You are then prompted for a number of communities to try. The tool attempts SNMP access to all the routers using each of the communities. If any routers are accessed successfully, these communities are entered in the authentication file to match the router names.

You can run the `auth_try_communities` tool repeatedly to add communities to the authentication file.

Note There is no equivalent tool for SNMPv3.

View Authentication Information

You can view the entire contents of the authentication file using the `mate_auth_export` tool, which exports a decrypted version of an authentication file. You can also view authentication information for a specific router using the `mate_auth_test` tool. Either way, you need the master password to view the contents.

Test the Authentication File

Test the authentication file using one of these tools.

- `mate_auth_test`—Prints authentication credentials for a specified authentication file, for a specified node IP address. The output returns whether the lookup is successful or optionally, shows all authentication details in plain text.
- `snmp_test`—Tests access to a specified router by sending a ping and an SNMP query using the credentials in the authentication file. If both SNMPv2c and SNMPv3 are present, then SNMPv3 is used.

- `login_test`—Tests login access to a specified router; in doing so it tests the login information provided by the authentication file.

Manage Archives

Note The tasks of configuring a plan file archive repository and inserting plan files into the archive are supported in both augmentation and manual collection methods. If you are collecting data only by configuring the WAE Collector web UI, then the archives described in this chapter are not applicable.

An *archive* is a repository containing network plan files, specific data items for plotting, and other data that is collected through augmented or manual snapshots. Additionally, information can be added to archives using CLI tools outside the snapshot process.

This chapter describes the basic archive tools.

Create or Update an Archive

Use `archive_init` to either create a new archive repository or to update the file structure of an existing archive.

- To create a new archive, set the `archive_init -archive` parameter to the path and name of the directory that will hold the archive. This creates a new, empty archive. The structure and support files for the archive are not complete until after the first recorded insertion.
- To update the file structure of an existing archive to that of the latest release, set the `archive_init -archive` parameter to the directory of an existing archive and set `-upgrade` option to `true`. This updates the file structure of the existing archive to that of the latest release.
- **For manual configurations of WAE Live**, you must override the default data typically extracted for an archive in order to create the Events panel.

```
archive_init -archive <Map Archive Path> -timeplot-summary-format
$CARIDEN_HOME/.cariden/etc/matelive/default_timeplot_summary_format.txt
```

Update Summary of Time-Sequence Plot Data

Use `archive_update` to update the summary of time-sequence plot data stored in an archive, for example after changing the summary format file.

- Set the `-archive` parameter to the location of an existing archive.
- Set the `-timeplot-summaries` parameter to `true`.
- Set the `-start-time` parameter to the timestamp of the first record to update.

By default, this tool updates all records from the start time stamp to the end of the archive, however, you can optionally specify the `-end-time`.

For information on configuring the time-sequence plot data, see the *Cisco WAE Design Archive User and Administration Guide*.

Insert or Extract Files from an Archive

Each archive record can contain one or more of the following files.

- Network plan file (.pln) obtained using the `snapshot` tool.
- Time-sequence plot summary file (.sum), automatically constructed using default summary format settings. This file can also be constructed and inserted manually. (See the *Cisco WAE Design Archive User and Administration Guide*.)
- Optional: User file or any other file.

For WAE Design Archive, the archive can also contain a visual format file, which specifies how the time-sequence plot data should be displayed in the web browser.

Insert Files

You can insert files all at once using one `archive_insert` tool, or individually using multiple CLI tools. All files in the archive repository are stored and accessed using a timestamp, so unless you want to use the default current timestamp when adding files to the archive, you include the `-time <timestamp>` option.

Note If the `snapshot` tool is configured to generate .txt format plan files, use the `mate_convert` tool before `archive_insert` to convert the .txt format plan file to the .pln format plan file.

Inserting a plan file into the archive automatically updates the files needed for interacting with the archive information via the web browser. For this reason, do not copy a file into the archive directory.

You can also insert WAE Design Archive plan files into the archive by choosing **File > Save to > Design Archive** in the WAE Design GUI. For information, see the *Cisco WAE Design Archive User and Administration Guide*.

Extract and Delete Files

After files have been archived, you can retrieve a copy using the `archive_extract` tool. CLI options specify which files to retrieve, where to copy them, and what to name them. You must include a timestamp. However, you can also specify that WAE Collector use the closest time to the timestamp provided, or you can specify a range of time to get a batch of files.

To retrieve user files with `archive_extract`, follow one of these options.

- Specify the name of the file to extract, or a partial name with wildcards (*), with the `-user-files` option.
- Specify a list of file names with the `-user-files-list` option.

You can also use `archive_extract` to remove items from the archive. The procedure is the same as for extracting files, except that you use the `-delete` parameter to delete the file after extraction. This process ensures that you always have a local copy of files that you delete, in case the deletion was accidental or incorrect.

You can also retrieve plan files from the archive by choosing **File > Open from > Design Archive** or **File > Open from > WAE Live** in the WAE Design GUI. For information, see the *Cisco WAE Network Visualization Guide*.

Manage Archives for WAE Design Archive

The `archive_config` tool enables you to manage the archive repositories available to the WAE Design Archive application on the web server. This tool creates an `archivelist.xml` file in the `$CARIDEN_HOME/etc/archive/config` directory.

- `-action add`—Add the archive repository to a specific location.
- `-name`—Name of the archive repository.
- `-path`—Full path of the archive repository.
- `-template-dir`—Full path of the template.
- `-template-name`—Name of the template used by all files in this archive repository.

Example: This example adds an archive named SW_Region that has a path of `acme/archives/acme_backbone`. The template directory is `acme/data` and the template name is `acme_backbone-template.pln`.

```
archive_config -action add -name SW_Region -path acme/archives/acme_backbone -template-dir
acme/data -template-name acme_backbone-template.pln
```

Note The `archive_config` CLI tool and the WAE Design GUI Archive feature do not apply to WAE Live.

Make Batch Changes to Archive Files

Maintenance of archives sometimes requires similar updates to multiple files in an archive. Here are two examples:

- A change in topology requires application of a new template file to the plan files between two time stamps
- A change in reporting requirements requires an updated summary file for plans for all plans in the archive.

You can perform this task with individual CLI tools, or in many cases you can use the `archive_do` tool to consolidate the CLI tools. The `archive_do` tool gets a list of timestamps between `-time` and `-time-to`, using `archive_extract`, and then performs the following for each timestamp.

- Uses `archive_extract` to extract all `%extract_*` files into a local directory.
- Executes CLI tools in `-cmd` argument sequence in the local directory. Table 4-10 lists the valid variables in the `-cmd` argument.
- Uses `archive_insert` to insert all `%insert_*` files into the archive.

The `archive_do` tool creates a list of CLI calls for all timestamps, fills in the temporary files at each step, and surrounds the calls with the relevant `archive_extract` and `archive_insert` tools. You can view the CLI tools without applying them to an archive by specifying the `-dry-run` option.

Table 4-10 Valid Variables for the -cmd Argument of archive_do

Variable	Description
%extract_plan	Plan file to extract at a given timestamp.
%extract_summary	Summary file to extract at a given timestamp.
%extract_user{FILENAME}	User file FILENAME to extract at a given timestamp.

Table 4-10 Valid Variables for the -cmd Argument of archive_do

Variable	Description
%insert_plan	Plan file to insert at a given timestamp.
%insert_summary	Summary file to insert at a given timestamp.
%insert_user{FILENAME}	User file FILENAME to insert at a given timestamp.
%timestamp	Current UTC timestamp, YYMMDD_HHMM.
%extract_previous_plan	Plan file extracted at previous timestamp; if this is first timestamp in archive, then the entire command is skipped for this timestamp.
%cariden_bin	Location of the binary files; this is useful if there is no path set. **Example:** %cariden_bin/table_extract

Example: This shows how to update the summary file for every plan in an archive.

```
archive_do -archive /opt/archives/my_archive -cmd "table_extract -plan-file %extract_plan
-out-file temp.txt; table_extract -plan-file %extract_previous_plan -out-file
temp_previous.txt; mate_summary -table-file temp.txt -old-table-file temp_previous.txt
-summary-format-file new_format_file.txt -out-file %insert_summary”
```