



## **Cisco WAE Design 7.0.1 Integration and Development Guide**

First Published: 2017-09-12

Last Updated: 2017-09-12

### **Cisco Systems, Inc.**

[www.cisco.com](http://www.cisco.com)

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco website at [www.cisco.com/go/offices](http://www.cisco.com/go/offices).

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED "AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: [www.cisco.com/go/trademarks](http://www.cisco.com/go/trademarks). Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

© 2019 Cisco Systems, Inc. All rights reserved.



---

**CHAPTER 1****Overview 1-1**

Integration with Other Systems and Workflows 1-1

---

**CHAPTER 2****Plan Files and Tables 2-1**

Plan Files 2-1

.pln Format Plan 2-1

.txt Format Plan 2-1

Convert Between Plan File Formats 2-2

Plan Tables 2-2

Plan Table Columns 2-2

Table Objects 2-3

Table Schema and Plan Tables 2-4

Working with Tables 2-4

Edit Plan File Tables Using CLI Tools 2-5

Edit Plan File Tables Using the GUI 2-7

Error Handling When Opening Edited Plans 2-8

Tables as Command Arguments 2-8

SQL Queries in WAE Design 2-9

User-Defined Columns and Tables 2-10

Create User-Defined Columns 2-11

Create User-Defined Tables 2-12

Change Column Appearance 2-13

Create Filter Interactions in Tables 2-14

External Tables 2-19

Demand Mesh Table 2-20

Edits Table 2-20

---

**CHAPTER 3****Importing Objects 3-1**

Import SRLGs 3-1

Import Layer 1 3-2

Import QoS Model 3-2

Import Demand Groupings 3-3

Import External Endpoints 3-4

Import Tags 3-5

---

**CHAPTER 4**

**Importing Traffic and Growth Rates 4-1**

Traffic and Growth Rate Imports 4-1

Import Rules 4-1

Traffic Tables 4-2

Growth Tables 4-4

Import Traffic and Growth Rates 4-5

Demand Grouping Traffic 4-6

<DemandGroupingTraffic> Table 4-7

Import Demand Grouping Traffic Growth 4-9

Representative Plan Files 4-9

Examples 4-11

---

**CHAPTER 5**

**Exporting Routes 5-1**

Export Routes 5-1

Example Export Route Tables 5-2

Export Routes Using the GUI 5-3

Export Explicit LSP Path Settings 5-4

Export L1 Link Wavelength Utilization 5-5

Export Current Table 5-6

---

**CHAPTER 6**

**Importing Offline Collections 6-1**

Import Router Configuration Files 6-1

Import IGP Database 6-3

---

**CHAPTER 7**

**Add-Ons and GUI Customizations 7-1**

Add-On Applications 7-1

Create Add-Ons 7-1

Customized Network Plot Views 7-8

Map Server 7-8

Static Backgrounds 7-8

---

**CHAPTER 8**

**Reporting Tools 8-1**

Reports in Plan Files 8-1

Plan File Report Format 8-1

Example Plan File Report Directory and Tables 8-2

---

**CHAPTER 9****Command-Line Interface 9-1**

CLI Options File 9-1

Logging Levels 9-2





## Overview

---

Use this guide if you are integrating the WAE Design application and doing related developmental work, such as creating custom add-ons. This guide includes the following topics:

- [Plan Files and Tables](#)—Describes plan file formats and how information is stored in tables. It also describes how to create user-defined columns and user-defined tables.
- [Importing Objects](#)—Explains how to import Layer 1 (L1) models, SRLGs, QoS models, demand groupings, and tags from other plan files.
- [Importing Traffic and Growth Rates](#)—Explains how to import traffic and growth values into plan files, and describes the tool that lets you create representative plan files by aggregating a series of plans over a time period.
- [Exporting Routes](#)—Explains how to export the routes of demands, IGP shortest paths, LSPs, LSP paths, circuits, and L1 circuits. It also describes how to export LSP explicit path settings, export L1 circuit Lambda Sim values, and how to export the current table showing in the WAE Design GUI.
- [Importing Offline Collections](#)—Describes the WAE Design GUI tools for importing router configuration files and IGP databases into plan files.
- [Add-Ons and GUI Customizations](#)—Explains how to create add-on applications that can be accessed from the WAE Design GUI, as well as other means of customizing the GUI.
- [Reporting Tools](#)—Describes tools for managing reports integrated into plan files.
- [Command-Line Interface](#)—Describes the file for setting default options used by calls to the tools through the CLI or through the GUI, as well as options that control the level of CLI logging.



### Note

This guide references `$CARIDEN_HOME`, which is the directory in which the Cisco WAE executables and binaries are installed. On Linux, the default `$CARIDEN_HOME` is `/opt/cariden/software/mate/current`, where `/opt/cariden` is the default installation directory.

## Integration with Other Systems and Workflows

The information in a plan file is readily available for integration into other systems or workflows. All plan information is saved in tables, which you can retrieve, filter, and export to ordinary .txt format plan files. The easiest way to become familiar with the tables in a plan file is from the GUI, using the Plan Table Database Editor. This interface displays a list of all the tables in a plan and provides options for extracting plan information, from simple text searches to complex SQLite queries. The data you extract can be useful for critical tasks, such as:

- Creating reports about utilization and routing for a single simulation scenario or multi-scenario analysis.
- Extracting paths for node pairs, specific demands, or LSPs for analysis by third-party applications.

If you have a custom task that is part of your workflow, you can add that task as a menu item to the WAE Design GUI, using the Add-on capability. Add-ons can simplify the task of data extraction for other systems, as well as for any WAE Design task that you perform frequently.



## Plan Files and Tables

---

This chapter provides information about the tables used in WAE plan files.

### Plan Files

*Plan files* contain a complete representation, or model, of a network. This includes topology, configuration, state, traffic, and visualization data.

WAE creates plan files when collecting from a network and makes them available for use. CLI tools and WAE Design GUI tools also provide a means of creating and manipulating plan files.

Plans have the following file formats:

- *.pln* format plan—Complete plan information in the native WAE format.
- Complete *.txt* format plan—Complete plan information in a *.txt* format plan file.
- Simple *.txt* format plan—Simplified plan information in a *.txt* format plan file.

The GUI and CLI tools use these formats interchangeably for input and output files.

### *.pln* Format Plan

The *.pln* format is the native WAE format for plan files. This format is small in size, so opening and saving the file is fast. When scripting, you generally work with some combination of *.pln* and *.txt* format plans and tables.

### *.txt* Format Plan

Each *.txt* format plan contains plan information in plain text that you can view and edit in an Excel spreadsheet or text editor. Excel is particularly useful because most of the data has tab-delimited fields that Excel can convert to a spreadsheet.

Each file contains a collection of tables, including the required `<Network>` table that specifies the WAE version that created the plan and other high-level plan properties. Each table is preceded by a table heading, such as `<Nodes>` and `<Sites>`. Rows that start with a pound sign (`#`) are ignored and treated as comments.

There are two types of .txt format plan files:

- Complete—Contains all of the schema of tables and columns described in [External Tables](#). Scripts written on complete .txt format plan files do not have to verify columns and tables are present before operating on them.
- Simple—Contains a subset of the complete format, eliminating the following unnecessary tables and columns:
  - All empty tables.

Example: <LSPs> table with no entries.

- All columns that contain only default values.

Example: Empty Protected values in the <Nodes> table default to F. So if all values in the column are F, it is not included in the .txt format plan.

This format is smaller and easier to edit manually. However, it can be more difficult for a script to parse because the script must check for missing tables and columns. We do not recommend this format when using scripts.

## Convert Between Plan File Formats

The GUI and CLI use the .pln and .txt formats interchangeably. When you save a file from the GUI, the .pln format is the default. To save to a different format, choose **File > Save as**.

From the CLI, the format of a saved file depends on the extension of the output filename. A .pln extension produces a .pln format file and a .txt extension produces a .txt format file. A command-line argument (`-simple-txt-out-file`) specifies the simple .txt format plan.

To convert one format to another, use the `mate_convert` CLI tool. This tool can convert formats within the current software version, or from an earlier version to the current version.

## Plan Tables

All aspects of a plan are defined using a collection of tables. Developers can access and modify all tables using command-line tools. For a complete listing of tables and their columns, see the `$(CARIDEN_HOME)/docs/table_schema.html` file.

## Plan Table Columns

The columns in a plan file table contain data in one of the following categories:

- Key columns—Columns that uniquely identify the rows of the table. Each table has one or more key columns. For example, the <Nodes> table has one key column, Name, which is the unique name of the node. The <Interfaces> table has two key columns: Node and Interface. This pair must (jointly) be unique for all entries in the table. Another example is the <Demands> table, which has key columns: Name, Source, Destination, and ServiceClass. If there are two demands from the same source to the same destination with the same service class, they must have different names.

In the `table_schema.html` file, key columns are highlighted in orange.

- Plan columns—Columns that define or configure properties of entries in a table. For example, in a <Nodes> table, the Site column specifies the site that contains the node, and is therefore a plan column. Key columns are always plan columns.

In the `table_schema.html` file, plan columns are highlighted in blue.

- **Derived columns**—Columns that provide information that is derived from the plan columns in the same table or a different table. These are not stored in plan files, but are generated by the GUI when the tables are displayed, or by `table_extract` when the tables are extracted from the plan file. For example, Remote Node in the <Interfaces> table is derived by looking up the remote node for the interface as defined in the <Circuits> table. Some derived information can be more complex to obtain. For example, the Traff Sim column is a derived column that is the result of a simulation performed on the network.

The entries in tables generated in the GUI and from `table_extract` can depend on some pre-specified parameters. For example, the <Interfaces> table Traff Meas column is the measured traffic on that interface for a specified traffic level. For a particular QoS selection the column can be overall (*undifferentiated*) traffic, traffic on a particular queue, or traffic for a particular service class.

In the `table_schema.html` file, derived columns are highlighted in gray.

## Table Objects

Objects in WAE Design are represented by rows in tables, and object properties are represented by column entries in that table, or by entries in tables of related objects. For example, LSP objects are defined in the LSPs table. Columns in this table, such as Setup BW, define properties of each LSP. The paths of each LSP are also properties of the LSP, but those LSP paths are defined as objects in separate LSP Paths, Named Paths, and Named Path Hops tables.

Table 2-1 lists the notation that WAE Design uses to specify a plan object when the type of object is not known from the context. Except for the IP address, these notations have a one-to-one mapping with key columns for each object.

**Table 2-1** Single Object Notation

Object	Format
AS	AS{ASN}
Circuit	ct{NodeA   InterfaceA   NodeB   InterfaceB} Example: ct{atl   POS1/1/1   sjc   POS1/10}
External endpoint	EP{Endpoint} Example with a member: EP{100}:cr1.chi
Interface	if{Node   Interface}
IP address	ip{ipaddress} Can reference multiple objects of the same or different type. WAE first tries to find a node with this loopback IP. If it is not found, WAE looks for a matching interface. If there are multiple matches, WAE uses the first one it finds.
Layer 1 circuit	l1ct{L1NodeA   L1NodeB   Name}
Layer 1 circuit path	l1ctp{L1NodeA   L1NodeB   L1Circuit   PathOption}
Layer 1 link	l1lnk{L1NodeA   L1NodeB   Name}
Layer 1 node	l1nd{Name}
Layer 1 port	l1pt{L1Node   L1Port}
Node	nd{Name}

**Table 2-1** Single Object Notation (continued)

Object	Format
Port	pt{Node   Port}
Port circuit	pct{NodeA   PortA   NodeB   PortB}
Site	st{Name}
SRLG	srlg{Name}

## Table Schema and Plan Tables

In a text plan file, each table starts with <TableName> to identify the name of the table; for example, <Nodes>.

The first row of the table body contains the column headings, followed by rows that describe the table entries. The order of the columns is irrelevant, and only the key columns must be present.

Each plan table is defined using an excerpt from the database schema, the part that defines that table. For example, [Table 2-2](#) lists a table schema excerpt for the <NamedPaths> table. [Table 2-3](#) shows an example of a <NamedPaths> table that has been populated within a plan file.

In [Table 2-3](#), the first column is Name, which is described in the first row of [Table 2-2](#). In this case, the Name of the named path (PathA or PathB) is the same in the plan file and GUI, it's the name of the path, has a data type of text, and is a table key. Being a key means the Name column is among the columns that uniquely define a row. In this case, the Name and Source together define a unique row.

**Table 2-2** NamedPaths Table Schema

Plan Filename	User Interface Name	Description	Data Type	Category
Name	Name	Name of the path	text	key
Source	Source	Name of the source of the NamedPath	text	key
Active	Active	Is the path active?	boolean	plan
Resolved	Resolved	T if all hops in path are resolved in plan, F if not, na if no hops.	boolean	derived

**Table 2-3** <NamedPaths> Example

Name	Source	Active	Resolved
PathA	Router1.AcmeG	T	T
PathB	Router1.AcmeO	T	T

## Working with Tables

This section describes how to modify plan tables from the CLI or GUI, or how to use table files as arguments for command-line tools.

## Edit Plan File Tables Using CLI Tools

You can add, modify, or delete tables in a plan file. Basically, you extract the desired table, edit the extracted file, then replace the table in the plan. Changes to columns of type Derived are ignored when replacing a table. Only columns of type Plan are relevant for changes.

The replacement tool does not validate the changed table, so the resulting plan could be incomplete or incorrect. Such problems are flagged the next time the plan is opened in WAE Design.

### Add or Delete Table in a Plan

- Add the table to the plan using `table_replace`. This overwrites an existing table if one exists.
- Delete a table from a plan using `table_delete`.

### Change Table in a Plan

**Step 1** Extract the table from the plan using `table_extract` and save it to a .txt format file.

**Step 2** Change the data in the table using in one of the following tools:

- Apply changes with a simple SQL statement by using `table_edit`.
- Apply changes with a complex SQL statement by using `mate_sql`.
- Apply changes manually using Excel or a text editor.

**Step 3** Replace the table in a plan file with the modified table by using `table_replace`.

You normally replace the table in the original plan file, but the result of `table_replace` can be a different file or even a different plan format. For example, you could replace a table in a text format plan and save the result as a binary format file (.pln), leaving the original file unchanged.

### Example of a Table Modification Using CLI Tools

This example shows how to increase the IGP Metric for interfaces on core routers in the `us_wan` plan, which is in the `$CARIDEN_HOME/samples` directory. [Figure 2-1](#) shows an excerpt from the original plan file. The Interfaces table has three columns: Node, Interface, and IGPMetric, which are all either “key” or “plan” columns. You only need the “key” values to uniquely define an interface.

**Figure 2-1** Interfaces Table in `us_wan.txt`

	A	B	C	D	E
91	<Interfaces>				
92	Node	Interface	IGPMetric		
93	cr1.atl	{to_cr1.hst}	37		
94	cr1.atl	{to_cr2.atl}	1		
95	cr1.atl	{to_cr2.mia}	36		
96	cr1.atl	{to_er1.atl}	1000		
97	cr1.bos	{to_cr2.bos}	1		

381795

In this simple example, you could manually update the IGPMetric column, using Excel or another text editor. To script the process, use CLI tools.

**Step 1** Extract the Interfaces table from the plan file to a temporary file, `if-table.txt`.

```
table_extract -plan-file us_wan.txt -out-file if-table.txt -tables Interfaces
```

**Figure 2-2** *Extracted Interfaces Table*

	A	B	C	D	E	F	G
1	<Interfaces>						
2	Node	Interface	RemoteNode	RemoteInterface	Function	IGPMetric	TraffSim
3	cr1.atl	{to_cr1.hst}	cr1.hst	{to_cr1.atl}	core	37	752.74
4	cr1.atl	{to_cr2.atl}	cr2.atl	{to_cr1.atl}	core	1	393.57
5	cr1.atl	{to_cr2.mia}	cr2.mia	{to_cr1.atl}	core	36	52.16
6	cr1.atl	{to_er1.atl}	er1.atl	{to_cr1.atl}	edge	1000	209.55
7	cr1.bos	{to_cr2.bos}	cr2.bos	{to_cr1.bos}	core	1	0

Figure 2-2 shows an excerpt from the extracted Interfaces table. The interesting part of the extracted table is that it contains more than just “key” and “plan” columns, it also has “derived” columns, which WAE Design creates when it reads a plan file or performs simulations. These extra columns make it easier to identify the core routers in this example because there is now a Function column.

**Step 2** The next step is to increase the IGP Metric for core router interfaces. The command below reads the extracted Interfaces file, `if-table.txt`, finds the IGPMetric column of the Interfaces table, filters the rows to those with ‘core’ in the Function column, adds 100 to the IGPMetric in the filtered rows, and saves the result to a new file named `if-table-edited.txt`.

```
table_edit -plan-file if-table.txt -out-file if-table-edited.txt -table Interfaces -column IGPMetric -rowfilter "Function = 'core'" -value IGPMetric+100
```

**Figure 2-3** *Interfaces Table After Replacement*

	A	B	C	D	E	F	G
1	<Interfaces>						
2	Node	Interface	RemoteNode	RemoteInterface	Function	IGPMetric	TraffSim
3	cr1.atl	{to_cr1.hst}	cr1.hst	{to_cr1.atl}	core	137	752.74
4	cr1.atl	{to_cr2.atl}	cr2.atl	{to_cr1.atl}	core	101	393.57
5	cr1.atl	{to_cr2.mia}	cr2.mia	{to_cr1.atl}	core	136	52.16
6	cr1.atl	{to_er1.atl}	er1.atl	{to_cr1.atl}	edge	1000	209.55
7	cr1.bos	{to_cr2.bos}	cr2.bos	{to_cr1.bos}	core	101	0

Figure 2-3 shows an excerpt from the updated file, `if-table-edited.txt`, which has the expected result in the IGPMetric column: the four core routers (nodes) shown have their IGP metrics increased by 100, and the edge router metric is unchanged.

**Step 3** The last step is to update the original plan file, or to create a new one. In this case, the example command creates a new plan file, one that uses the `.pln` format. Changing the file format in this example shows the flexibility of the tools.

```
table_replace -table-file us_wan.txt -replace-table-file if-table-edited.txt -out-file us_wan.pln
```

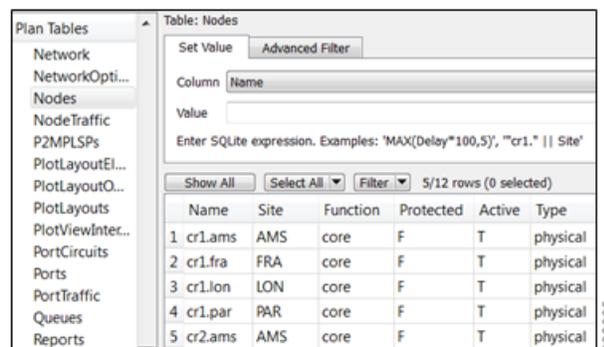
**Step 4** To verify the updates, open the `us_wan.pln` file. The GUI table contains the information shown in Figure 2-3.

## Edit Plan File Tables Using the GUI

Although you can edit plan files in the WAE Design GUI, we recommend that you edit them in a text editor or spreadsheet so that you can easily make global changes. A spreadsheet, such as Excel, is useful because it clearly shows table columns in a tabular format.

Any time you make a change in the WAE Design GUI and save the plan file, you are, in effect, changing the tables in the plan. Additionally, you can choose **Edit > Plan Tables as Database** in the WAE Design GUI to view and edit tables directly in the plan file. In this dialog box, you can select one or more rows to modify attribute values in the columns. You can specify the desired selection from the Advanced Filter tab, by entering SQLite queries, and specify value changes in the Set Value tab, also using SQLite syntax.

Values defined in one table might be referenced in other tables. Be careful when changing a value that occurs in multiple places. For example, the Nodes table identifies the sites in which nodes reside. If you change a site name in the Sites table, you must also change it in the Nodes table and all other instances.



- Step 1** Choose **Edit > Plan Tables as Database**. The Plan Table Database Editor opens, showing all possible tables in the left pane.
- Step 2** In the left pane, select the desired table. If the table already exists in the plan, it is displayed.
- Step 3** Optional: Filter the table entries by using the Filter control menu or by entering text in the Search field (top right).
- Step 4** Click the field you want to change and then edit the information, or choose one of these more advanced editing options.
  - Click the **Set Value** tab, choose the Column to modify, enter an SQL Lite expression that specifies the change, and click **Set**.
  - Click the **Advanced Filter** tab, enter an SQL Lite expression that specifies the change, and click **Filter**.
- Step 5** Visually verify that the change is correct.
- Step 6** To open the new plan, click **Open in WAE Design**. To save the new plan without opening it, click **Save as File**.

## Error Handling When Opening Edited Plans

Because plan files can be edited directly by the user, errors must be handled when opening plan files. The default behavior for table errors is to log a warning and ignore any row of any table that is inconsistent with information already processed. For example, if the <Nodes> table lists two nodes with the same name (which is a key column) the second row is ignored.

To simplify user editing of .txt format plan files, WAE Design gracefully handles common problems encountered during the opening process. For example, although the simplest plan can contain nodes, interfaces, and circuits, there are certain situations in which only an <Interfaces> table need be specified.

WAE Design makes the following changes when opening a plan with missing or incomplete information:

- Omitted tables are assumed to be present with no entries.
- Key columns are required for each row in a table. Other columns can be entirely omitted, and default values are assumed for all their entries. The default values are listed in the Table Format Reference.
- Individual entries that are left blank (in non-key columns) are filled in with default values.
- Any nodes created in the Node column of the Interfaces table are created in the <Nodes> table if they do not already exist.
- The columns Remote Node and Remote Interface in the <Interfaces> table are derived columns, which are normally ignored when WAE Design opens a plan. However, in this case, when the Node, Interface, Remote Node, and Remote Interface unambiguously identify another interface as the remote interface for this circuit, and the two interfaces are not already defined as belonging to any circuit in the <Circuits> table, then a circuit connecting these two interfaces is created.
- In the <Nodes> table, if the site entry is blank, a new site is created for this node with the same name as the node. Sites referenced in the <Nodes> table are added to the <Sites> table if not already present.
- If Node and Interface in the <InterfaceTraffic> table are both blank, and the (derived) column IPAddress in the table contains a valid reference to an interface in the <Interfaces> table, then (Node, Interface) for that interface is entered and the traffic is associated with that interface. This allows an <InterfaceTraffic> table to be added to the plan that references traffic measurements only by IP Address.
- If Source in the <LSPTraffic> table is blank, and the (derived) column SourceIP contains a valid reference to a node, and this Node and the Name entry uniquely identify an LSP, then that node name is entered into the Source column and the traffic is associated with that LSP.

## Tables as Command Arguments

Many CLI tools take arguments that require the specification of a set of plan objects. For example, `merge_nodes` takes a `nodes-table` parameter that specifies the list of nodes to merge. Such a list can be specified in a file containing a table in the WAE Design table format. Tables used as arguments must contain at least the key columns defined for that table for the tool to uniquely match the objects in the import table with the objects in the plan.

To create tables for use as command-line tools arguments, follow these steps:

- 
- Step 1** Extract the full table of objects from the plan that is the target of the command-line tool, using `table_extract`.

- Step 2** Use an editor to select the appropriate rows in the table, and delete the rest. Or, typically in a script, use `mate_select` to select the rows using SQL syntax.

## LSP Mesh Creation Example

To create an LSP mesh between the core routers in sites ATL, MIA, and WDC of the `us_wan.txt` plan, you must first create a file that contains the list routers in the mesh. The following example extracts the Nodes table from the plan file, and then selects the desired core routers, and saves the result as `lsp_nodes.txt`.

- Step 1** Extract the Nodes table from the plan file.

```
table_extract -plan-file us_wan.txt -out-file nodes-table.txt -tables Nodes
```

- Step 2** Select the desired core routers and save the result as `lsp_nodes.txt`.

```
mate_select -table-file nodes-table.txt -out-file lsp_nodes.txt -table nodes -filter
"(Site LIKE 'ATL' OR Site LIKE 'MIA' OR Site LIKE 'WDC') AND Name REGEXP '.*cr.*'"
-show-columns Name
```



### Note

The SQL query uses `LIKE`, rather than `=`, when selecting sites to avoid case-insensitivity problems.

The resulting `lsp_nodes.txt` file contains the following list of nodes:

```
<nodes>
Name
cr1.atl
cr1.mia
cr1.wdc
cr2.atl
cr2.mia
cr2.wdc
```

- Step 3** Create the LSP mesh by invoking the `lsp_mesh_creator` tool, specifying the `lsp_nodes.txt` file as a command-line argument.

```
lsp_mesh_creator -plan-file us_wan.pln -out-file us_wan_lsp.pln
-source-nodes-table lsp_nodes.txt -dest-equals-source true
```

## SQL Queries in WAE Design

Three command-line tools in WAE Design use SQL syntax for filtering, summarizing, and manipulating plan files and reports.

- `mate_select`—Filters tables in the reports.
- `mate_summary`—Summarizes tables in the reports, primarily to provide summary data for network visualization over time.
- `mate_sql`—An advanced SQL query tool.

WAE Design uses the SQLite implementation of the SQL language (see [www.sqlite.org](http://www.sqlite.org), especially [www.sqlite.org/lang.html](http://www.sqlite.org/lang.html)).

The following operators have special meaning in WAE Design:

**REGEXP**—Case-insensitive matching of regular expressions. For example, SQL expression

```
Name REGEXP '^cr'
```

Is true for Name equal to 'CR', 'Cr' or 'CR01'. (But not for Name equal to 'er.cr')

**MATCH**—Some columns in the tables contain semicolon-delimited lists, for example a list of tags in the Tags column of the Nodes and Interfaces tables, or the list of interfaces in the Actual Path column of the LSP table. The MATCH operator tests for membership in these lists. For example,

```
Tags MATCH 'Europe'
```

Is true for Tags equal to 'Asia;Europe', 'EUROPE', and so on. The matching is case-insensitive.

The operator '=' is case-sensitive in SQL. The operator LIKE, which is case-insensitive, is often more useful for plan schema tables because the case is never relevant.

- **SUBNET**—Selects rows if the fields look like IP addresses and are in a specific subnet.

This function has the following syntax options:

- SUBNET(Column\_Name, 'ip\_address/prefixlen')
- SUBNET(Column\_Name, 'ip\_address/netmask')
- SUBNET(Column\_Name, 'ip\_address', 'prefixlen')
- SUBNET(Column\_Name, 'ip\_address', 'netmask')

The following examples demonstrate usage of each syntax:

- SUBNET(Column\_Name, '192.168.1.0/24')
- SUBNET(Column\_Name, '192.168.1.0/255.255.255.0')
- SUBNET(Column\_Name, '192.168.1.0', '24')
- SUBNET(Column\_Name, 'ip\_address', '192.168.1.0, '255.255.255.0')

The following shows usage in a real SELECT statement:

```
select * from Table where subnet(IPAddress, '192.168.1.0/24');
```

This WHERE clause matches a row if the field IPAddress is an IP address in the 192.168.1.\* network.

SQLite does not allow arbitrary functions to have infix notation, so the following notation is impossible:

```
Where IPAddress SUBNET '192.168.1.0/24'
```

- **SUBSTITUTE**—Substitutes a new value for an old value in a column.

Syntax: SUBSTITUTE(Column\_Name, 'old', 'new')

where 'old' and 'new' are values like those used in the syntax: s/old/new

The following example shows how to replace the **m** with **c** in all node names that start with **mr**:

```
table_edit -plan-file x.txt -out-file y.txt -table Nodes -column Name
-value "SUBSTITUTE(Name, '(mr\).*', 'cr\1)'"
```

## User-Defined Columns and Tables

You can add new tables of your own design and new columns to the standard tables. You might also choose to create columns for use in tools such as the RSVP-TE Optimization (rsvp\_te\_opt) tool. Using these user-defined structures can help reduce the number of extra files required in a solution, and are helpful in custom scripts that need to carry information from one step to another.

Each user-defined table or column is prefixed with a *namespace*, which is a convenient way of grouping tables or grouping columns under one name to prevent conflicts with plan table names and plan columns.

WAE Design does not check user-defined columns and tables for errors. However, they are preserved on import and export, and their contents are displayed in the GUI and plan table hierarchy.

## Create User-Defined Columns

Once a user-defined column is added to a plan table, you can manipulate it in the WAE Design GUI. You can duplicate it, delete it, or edit its value by right-clicking the table row and clicking the **User** tab. You can also modify or delete user-defined columns by choosing **Edit > Plan Tables as Database**.

### Add Columns Using a Text Editor

- 
- Step 1** Open a plan file in a text editor.
- Step 2** Find the table to which you are adding the column.
- Step 3** Add a tabbed space at the end of the headings and then enter the column name using the following format. Note that all columns must have a tabbed space between them.
- ```
Namespace1::ColumnName
```
- You can chain Namespaces together using `::` between their names.
- ```
Namespace1::Namespace2:: ... ::ColumnName
```
- Step 4** For each object to which you want to enter a value, tab to the user-defined column, and enter the value.
- Example: [Table 2-4](#) shows a user-defined column that was added to a <Sites> table. The first three columns are schema columns. The last is a user-defined column where the Namespace is Customer and the ColumnName is Service. Combined, they form the user-defined column named Customer::Service. The values Voice and Video were entered on their respective rows.

**Table 2-4** Example: User-Defined Column

Name	Longitude	Latitude	Customer::Service
EMEA_SW	2.55	49.02	Voice
EMEA_Central	4.78	52.32	Video

---

### Add Columns Using the GUI

- 
- Step 1** Show the table in which you want to add the column.
- Step 2** Do one of the following:
- Choose **Edit > User Columns**.
  - Double-click a row in the table to open a Properties dialog box, and then click the **User** tab.
- Step 3** In the dialog box that opens, click **New**; a New User Property dialog box opens.
- a. Enter the prefix, name, and select the type (text, real, integer, or boolean).

- b. If you are creating this column through the User tab, you can enter a value for the selected property. Otherwise, to populate the column after adding it (after Step 4), you must open the Properties dialog box for the appropriate object and select the User tab to enter the value.
- c. Click **OK**.

**Step 4** Click **OK**.

---

## Create User-Defined Tables

Once a user-defined table is added, it is available in the WAE Design GUI and shows by default. Each user-defined table has a context menu containing Row Properties that lets you modify content directly from the table just like you would modify properties in plan tables. From this context menu, you can also delete or duplicate rows in tables. Duplications are exact, meaning, the duplicated rows do not contain objects that are renamed with the usual convention of [#].

**Step 1** Open a plan file in a text editor.

**Step 2** On an empty row, enter the table name using the following format. You only need one Namespace, though you can chain them together using :: between their names.

<Namespace1::TableName>

<Namespace1::Namespace2:: ... ::TableName>



**Note**

Do *not* use plan table names as Namespaces. The TableName can be the same as a plan table name if it is used with a unique namespace. For example, Sites::Test is not a valid table, but Test::Sites is.

**Step 3** Complete the table by adding user-defined columns as described in [Create User-Defined Columns](#).

Example: [Table 2-5](#) shows a new table that shows relationships between nodes and LSPs. The Namespace is LSP. The TableName is Relation. Combined, they form the user-defined table named <LSP::Relation>. (Note that singular “LSP” as a new namespace is permissible because the plan table name is the plural form “LSPs.”)

**Table 2-5** Example: User-Defined <LSP::Relation> Table

EastNode	WestNode	SRSID
cr1.ams	cr1.lon	77
cr2.ams	cr2.lon	14
cr1.fra	cr1.par	23
cr2.fra	cr2.par	14

If you are going to create filter interactions that operate on object types, the objects must use the object notation defined in [Table 2-1](#). [Table 2-6](#) shows an example in the Failure column. For information on creating object filters for user-defined tables, see [Create Filter Interactions in Tables](#).

**Table 2-6 Example: User-Defined <Objects::Failures> Table with Object Notation**

Failure	ServiceClass	TrafficLevel
nd{cr2.par}	Internet	peak
st{AMS}	Voice	weekend
ct{cr2.par to_er1.par er1.par to_cr2.par}	Undifferentiated	default

## Change Column Appearance

To define the column appearance for both user-defined columns and user-defined tables, use the <ColumnData> table, which uses the columns described in Table 2-7. Open a .txt file that is in a complete, tab-delimited format to see this table. Add one row per each user-defined column or table you are modifying.

Note that this table does *not* apply to plan columns in plan tables.

**Table 2-7 <ColumnData> Columns**

Column Name	Type	Description
Table	Key	The table to which this entry applies.
Column	Key	The column name (within the specified Table) to which this entry applies.
Type	Plan	The data type: Integer, Text, Boolean, and Real. If left empty, the type is Text.
Decimals	Plan	<p>If left empty and if the Type is Real, numeric values appear with two decimal places and the numbers are rounded upward.</p> <p>Example: The value in the user-defined column is 77. If the Decimals value is empty and the Type value is Real, in the WAE Design GUI this appears as 77.00.</p> <p>Example: The value in the user-defined column is 77.5678. If the Decimals value is empty and the Type value is Real, in the WAE Design GUI this appears as 77.57.</p> <p>If you enter a positive integer and the Type is Real, the number of decimals is carried over that amount.</p> <p>Example: The value in the user-defined column is 23.1237. If the Decimals value is “3” and the Type value is Real, in the WAE Design GUI this appears as 23.124.</p> <p>If left empty and the Type is Integer, all decimals are removed and there is no rounding.</p> <p>Example: The value in the user-defined column is 74.798. If the Decimals value is empty and the Type value is Integer, in the WAE Design GUI this appears as 74.</p>
DisplayName	Plan	<p>The column name displayed in the GUI if it differs from the name specified in Column. Since column names must be entered without spaces, this is useful for enhancing the display of two-word column names.</p> <p>If no tooltip is specified, the DisplayName is used as the tooltip.</p>
Tooltip	Plan	A tooltip for this column.
Shown	Plan	Controls whether a column is visible from the GUI by default. Values are T (True) and F (False). If left empty, the default is T (True).

Example: [Table 2-8](#) shows an example <ColumnData> table populated for the preceding two user-defined tables, and [Figure 2-4](#) shows how they appear in the WAE Design GUI.

**Table 2-8** Example: <ColumnData> Table

Table	Column	Type	Decimals	DisplayName	Tooltip	Shown
LSP::Relation	EastNode			East Node	Map to LSP source	True
LSP::Relation	WestNode			West Node	Map to LSP destination	True
LSP::Relation	SRSID	Integer		SR SID		True
Object::Failures	ServiceClass			Service Class		True
Object::Failures	TrafficLevel			Traffic Level		True

**Figure 2-4** Example User-Defined Tables in WAE Design GUI

LSP::Relation			
Show All	Select All	Filter	4/4 rows (0 selected)
East Node	West Node		SR SID
1 cr1.ams	cr1.lon	Map to LSP destination	77
2 cr2.ams	cr2.lon		14
3 cr1.fra	cr1.par		23
4 cr2.fra	cr2.par		33

Object::Failures			
Show All	Select All	Filter	3/3 rows (0 selected)
Failure	Service Class		Traffic Level
1 nd{cr2.par}	Internet		peak
2 st{AMS}	Voice		weekend
3 ct{cr2.par to_er1.par er1.par to_cr2.par}	Undifferentiated		default

407037

## Create Filter Interactions in Tables

You can configure filter interactions for report tables and for user-defined tables by creating a <ReportTableInteractions> table and <UserTableInteractions> table, respectively. These tables are the same except for the Key columns.

- <ReportTableInteractions>—Defines the filter interactions between the report tables and the GUI. Report tables are available by choosing **Window > Reports**.

All tools that return tables have this behavior defined, though you can edit the interaction behavior. User-created reports, such as those returned by add-ons, do not have these interactions by default. Note also that not all tools return tables in their reports.

- <UserTableInteractions>—Defines the filter interactions between user-defined tables and plan tables. Without defining these interactions, while the user-defined tables are available, there is no interaction between the user-defined table and the rest of the plan file data. For information on creating tables, see [User-Defined Columns and Tables](#).

The available filters are those that are already available from the plan file tables. For instance, in the L1 Nodes table, if you right-click an L1 node there are a number of contextual filters. If you were to create a filter interaction from a report table or user-defined table, and if that filter acted on L1 nodes, only those same L1 node filters would be available. You cannot further configure this filtering.

[Figure 2-5](#) shows a Simulations table in a Simulation Analysis report filtering to L1 nodes. Note that since the Summary section of this report is not a table, no filtering interactions can be configured for it.

**Figure 2-5** Simulations Table in Simulation Analysis Report Filtering to an L1 Node

	Failure	MaxUtil	MaxQoSBoundPercent	NumQoSVCs	NumUnrout
1	l1nd(kcy)				
2	st(kcy)				
3	l1nd(chi)	99.14		100.00	
4	st(chi)	99.14		100.00	
5	l1nd(sjc)	95.06		100.00	
6	st(sjc)	95.06		100.00	
7	l1nd(wdc)	56.42		100.00	
8	st(wdc)	56.42		100.00	

<ReportTableInteractions> and <UserTableInteractions> are configured in the same manner, as defined in [Table 2-9](#), except for the Key columns.

Table 2-9 &lt;ReportTableInteractions&gt; and &lt;UserTableInteraction&gt; Columns

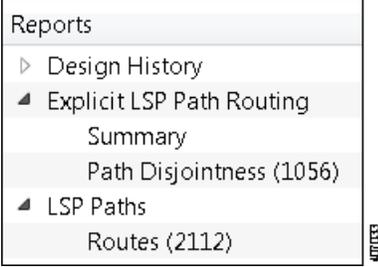
Interaction Table	Column Name	Type	Description
<ReportTableInteractions>	Report	Key	<p>Report name referenced in this entry. This must match the report named in the report output.</p> <p>Examples:</p> <ul style="list-style-type: none"> <li>If you run the Explicit Path Initializer, the resulting report is labeled “Explicit LSP Path Routing.” This is the exact entry you must use if modifying the filter interactions for this report. A report name of “Explicit LSP Routes” will not work.</li> <li>If you create an LSP path report using <b>Tools &gt; Reports &gt; LSP Path Routes</b>, it generates a report named “LSP Paths.” This is the exact entry you must use if creating filter interactions for this report. A report name of “LSP Routes” will not work.</li> </ul> 
<ReportTableInteractions>	Section	Key	<p>Report section name referenced in this entry. This must match the section named in the report output.</p> <p>Example: In the above two examples and figure, the viable Section names are Path Disjointness and Routes. The Summary section in the Explicit LSP Path Routing report is not tabular information and therefore, cannot be used.</p>
<UserTableInteractions>	UserTable	Key	<p>User table name referenced in this entry.</p> <p>Example: If the user-defined table on which you are defining interactions is named &lt;Object::Failures&gt;, that is the name you would enter here.</p>
Both	Reference	Plan	Name that appears as the selection when you right-click a row within the report table or user-defined table.
Both	ReferenceType	Plan	<p>Defines how the filter operates. Options are:</p> <ul style="list-style-type: none"> <li>Object—The user selection filters to the object type that is defined in the ObjectColumn.</li> <li>Table—The user selection filters to the table defined in the Table column.</li> </ul>

Table 2-9 &lt;ReportTableInteractions&gt; and &lt;UserTableInteraction&gt; Columns (continued)

Interaction Table	Column Name	Type	Description
Both	Table	Plan	<p>Required if the ReferenceType is Table.</p> <p>Identifies the name of the plan table to which you want to filter. Do not use spaces in the name.</p> <p>Example: The &lt;L1Links&gt; table is entered as L1Links. This means if you click any item in this report table or user-defined table, the filter selection is the same as if you were in the &lt;L1Links&gt; table.</p>
Both	SourceJoinColumns	Plan	<p>Required if the ReferenceType is Table.</p> <p>List of one or more comma-separated columns in the report table or user-defined table.</p> <p>Combined with the DestJoinColumns, this maps the selected row to the filtered table. There is a one-to-one, sequential mapping between the entries in each of these two columns. That is, the first entry for SourceJoinColumns maps to the first entry in the DestJoinColumn, the second entries map to each other, and so on until all entries are mapped. Therefore, the list must contain the same number of entries as DestJoinColumns. For a complete example, see <a href="#">Example: Filtering to Tables</a>.</p> <p>Example: If the second SourceJoinColumns entry is Active, the filter maps the Active column in the report table or user-defined table to the second entry (column) identified in DestJoinColumns.</p>
Both	DestJoinColumns	Plan	<p>Required if the ReferenceType is Table.</p> <p>List of one or more comma-separated columns in the report table or user-defined table that identifies the column to use in the table from which you are filtering.</p> <p>The list must contain the same number of entries as SourceJoinColumns. For a complete example, see <a href="#">Example: Filtering to Tables</a>.</p> <p>Example: If the DestJoinColumns entry is Traff Sim and the Table entry is Interfaces, the filter looks for the Traff Sim column in the &lt;Interfaces&gt; table. If this is the first entry in DestJoinColumns, it maps to the first entry in SourceJoinColumns.</p>
Both	ObjectColumn	Plan	<p>Required if the ReferenceType is Object.</p> <p>Identifies the column name in the report table or user-defined table that specifies the object type on which to filter. The objects in this column must be in the object-notation format listed in <a href="#">Table 2-1</a>. For an example, see <a href="#">Example: Filtering to Objects</a>.</p>

## Example: Filtering to Tables

[Table 2-5](#) shows a user-defined table named <LSP::Relation>. Notice that object definitions do not require object notation when using a ReferenceType of Table.

[Table 2-10](#) shows a <UserTableInteractions> table that defines the following:

- UserTable defines a table interaction filter for the <LSP::Relation> table in the WAE Design GUI.

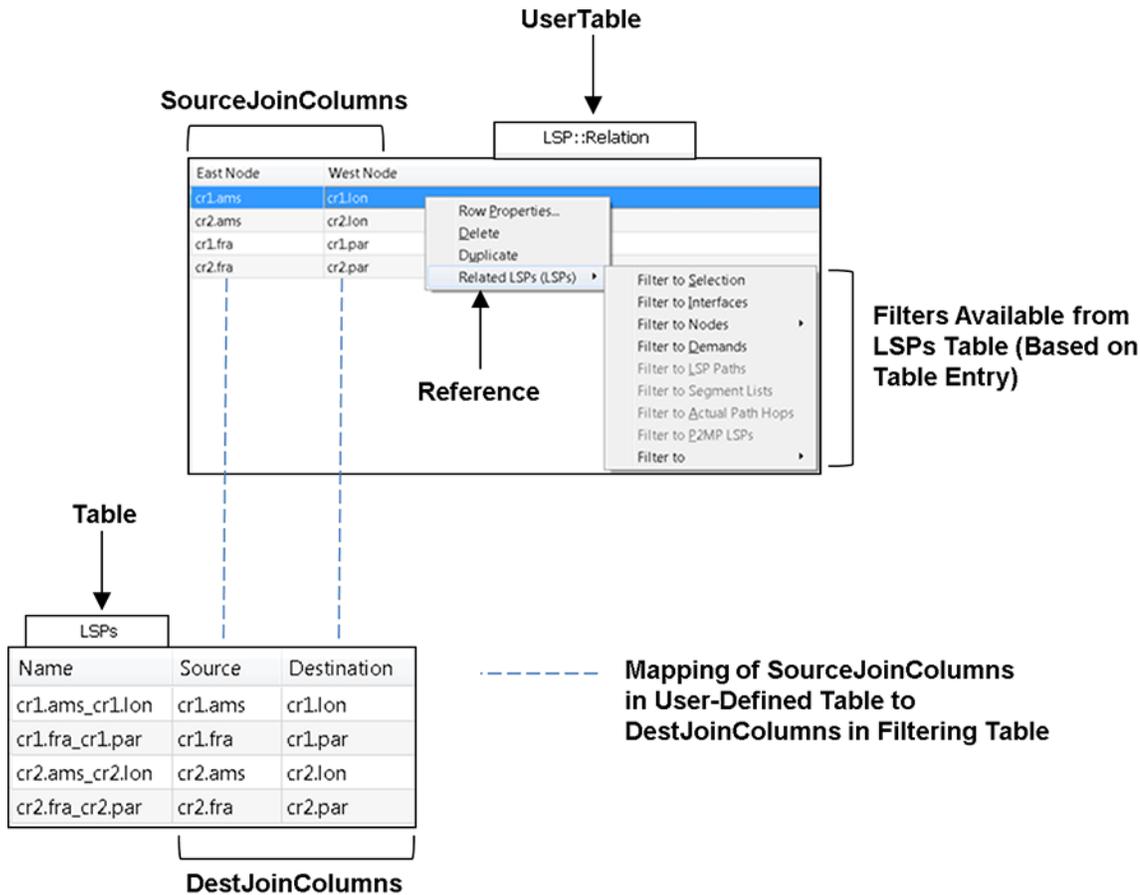
- ReferenceType is Table, so the filtering defined in this row explicitly references a plan table by name in the Table column.
- Reference defines that when you right-click a row in the <LSP::Relation> table, the context menu shows “Related LSPs.”
- Table defines that the filtering table is <LSPs>.
- To determine what to find in the <LSPs> table, the filter uses the SourceJoinColumns and DestJoinColumns to look for both of the following:
  - A node in the East Node column of the <LSP::Relation> table that is listed in the Source column of the <LSPs> table.
  - A node in the West Node column of the <LSP::Relation> table that is listed in the Destination column of the <LSPs> table.

If an entry in the <LSPs> table satisfies both conditions, the contextual filters for LSPs is made available to the selection as shown in Figure 2-6.

**Table 2-10 Example: <UserTableInteractions> Defining Two Table Filters**

UserTable	Reference	ReferenceType	Table	SourceJoinColumns	DestJoinColumns	ObjectColumn
LSP::Relation	Related LSPs	Table	LSPs	EastNode,WestNode	Source,Destination	

**Figure 2-6 Example Table Filtering**



407036

## Example: Filtering to Objects

Table 2-6 shows a user-defined table named <Objects::Failures>. Note that the objects (a node, site, and circuit) listed in the Failure column use the object-notation format described in Table 2-1.

Table 2-11 shows a <UserTableInteractions> table that defines the following:

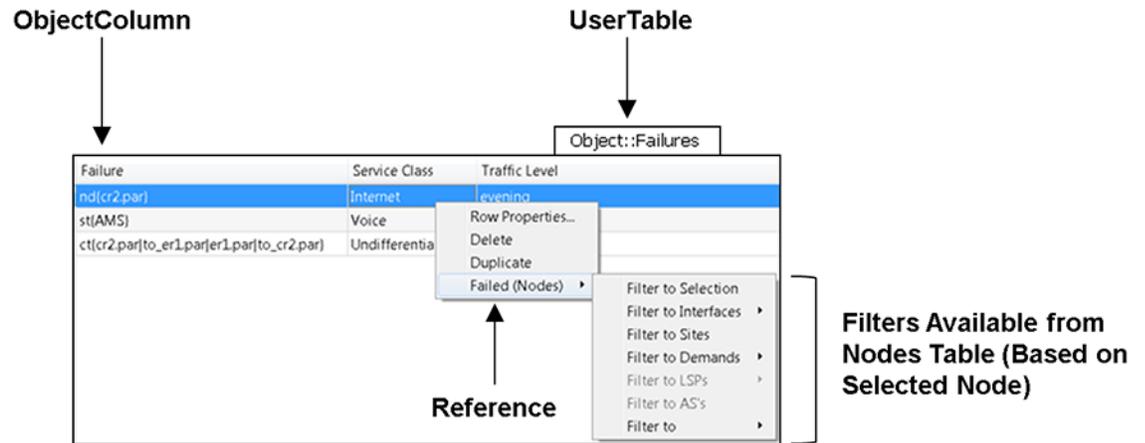
- UserTable defines a table interaction filter for the <Objects::Failures> table in the WAE Design GUI.
- Reference defines that when you right-click a row in the <Objects::Failures> table, the context menu shows “Failed.”
- ReferenceType is Object, so the tables from which the selections filter differ depending on which object is selected.
- ObjectColumn defines the column used to identify the object on which to filter.

In Figure 2-7, if you right-clicked on the first row and selected Filter to Selection, you would go to the node named cr2.par in the Nodes table. If you selected any other filter, you would go to the same filter as if you had selected cr2.par from the Nodes table. For instance, if you selected Filter to Sites, you would go to the same site in the Sites tables as if you had selected cr2.par from the Nodes table and filtered to it.

**Table 2-11** Example: <UserTableInteractions> Defining an Object Filter

UserTable	Reference	ReferenceType	Table	SourceJoinColumns	DestJoinColumns	ObjectColumn
Object::Failures	Failed	Object				Failure

**Figure 2-7** Example Object Filtering



## External Tables

In addition to plan tables, external tables provide input to plan files or are the result (output) of running tools on the plan file.

- [Demand Mesh Table](#).
- [Edits Table](#).
- Tables for importing traffic and traffic growth rates; see [Importing Traffic and Growth Rates](#).

- Tables for exporting routes and explicit LSP path settings; see [Exporting Routes](#).

## Demand Mesh Table

A <DemandMesh> table contains columns that identify the source and destination endpoints for a demand mesh, and optionally contains columns that specify the source and destination traffic for each (Figure 2-8).

- Endpoint—Represents the source/destination of a demand. For nodes, this corresponds to the node name. Other types of endpoints can also be specified. For example, 'AS{12345}' is a valid entry for an AS.
- SrcDest—Has a value of *Src*, *Dest*, or *both*, which specifies whether to use the endpoint as the source of the new demands, as the destination, or as both.
- SrcTraffic—Represents the source traffic (used by the `dmd_traffic_creator` tool).
- DestTraffic—Represents the destination traffic (used by the `dmd_traffic_creator` tool).

**Figure 2-8** Example <DemandMesh> Table

<DemandMesh>			
Endpoint	SrcDest	SrcTraffic	DestTraffic
cr1.ams	Src	107	204
cr1.fra	Dest	575	349
cr1.lon	both	367	437
cr1.par	both	136	386

381794

## Edits Table

The `table_edit` CLI tool can optionally use a file containing an <Edits> table, which is a very time-efficient means of globally modifying plan schema tables.

- Table—The name of the table to edit.
- Column—The name of the column in the table to edit. Must be one of the plan configuration columns, not derived/extended columns. If the column does not exist, it is created if the column is of the form `Namespace::ColumnName`.
- (RowFilter)—SQL WHERE statement selecting rows in the table. If empty, defaults to all. All columns, including derived and joined columns, are available for selection.
- Value—SQL Expression with value to insert in the column matching the filter selection. All columns, (including derived and joined columns, and including the column currently being edited, are available to use in this expression.

Example: This example shows an Edits file that change the forecast values in the demands table. The following tables show the original demands table, the edits table, and the updated demands table after running `table_edit`.

**Table 2-12** Original <Demands> Table

Source	Destination	GrowthRate
A	B	1

**Table 2-12** Original <Demands> Table (continued)

Source	Destination	GrowthRate
A	C	1
B	A	1

**Table 2-13** <Edits> Table

Table	Column	RowFilter	Value
Demands	GrowthRate	Source='A'	10
Demands	GrowthRate	Source='B'	20

**Table 2-14** Updated <Demands> Table

Source	Destination	GrowthRate
A	B	10
A	C	10
B	A	20





## Importing Objects

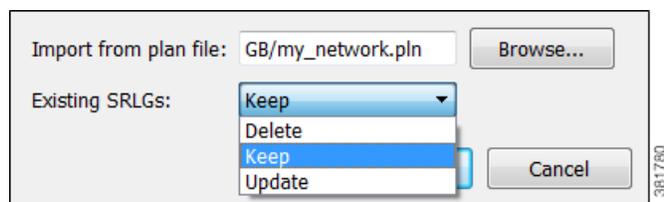
Creating a plan file for modeling purposes requires specifying a number of objects that cannot be discovered directly from the network. WAE Design lets you import shared-risk link groups (SRLGs), a Layer 1 (L1) model, a QoS model, demand groupings, external endpoints, and tags from one plan file to another, all from **File > Import**. You can also import demand groupings from a file containing a <DemandGroupings> table.

These import tools simplify and expedite the process of importing specific objects and models. For instance, if you want to copy an L1 model into an existing plan file, you could do so using the Copy from Template tool, which copies over more information than just the Layer 1 model. Using these import tools described in this chapter allows you to copy only what is necessary.

### Import SRLGs

When importing SRLGs, you can combine the imported SRLGs with existing SRLGS as follows.

- **Delete**—Delete all existing SRLGs in the destination plan file. Copy all SRLGs from the source plan file.
- **Keep**—Do not delete SRLGs in the destination plan file. Copy only SRLGs from the source plan file that do not have the same name as SRLGs existing in the destination plan.
- **Update**—Do not delete SRLGs in the destination plan file. Copy all SRLGs from the source plan file. If the two plans have SRLGs of the same name, the SRLG in the destination plan is replaced.



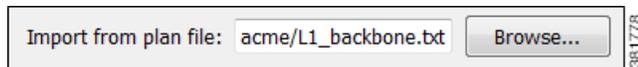
- 
- Step 1** Choose **File > Import > SRLGs**.
- Step 2** Browse to or enter the full path of the plan file from which you are importing SRLGs.
- Step 3** Select how you want the existing SRLGs handled (Delete, Keep, Update), and click **OK**.
-

For More Information...	See...
SRLGs	<i>Cisco WAE Design User Guide</i>
import_srlgs CLI tool	import_srlgs Help Output

## Import Layer 1

Importing a Layer 1 model imports all Layer 1 objects, including L1 nodes, L1 links, L1 circuits, L1 circuit hops, L1 ports, and L1 waypoints from one plan file to another.

- All L1 objects in the destination plan file are deleted and replaced by the newly imported L1 objects.
- If an L1 node exists in a site that also exists in the destination plan file, it is placed in that same site.
- If an L1 node exists in a site that does not exist in the destination plan file, the imported site is named after the imported L1 node.
- If the source plan file contains mappings between L3 circuits and L1 circuits, and if the destination plan file contains L3 circuits with the same key column definitions, these mappings are imported. The key columns are Node A, Interface A, Node B, and Interface B.



**Step 1** Choose **File > Import > Layer 1**.

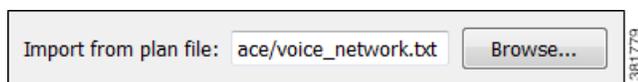
**Step 2** Browse to or enter the full path of the plan file from which you are importing L1 objects, and click **OK**.

For More Information...	See...
Layer 1 objects and Layer 1 network simulation	<i>Cisco WAE Design User Guide</i>
import_layer1 CLI tool	import_layer1 Help Output

## Import QoS Model

The QoS model consists of the mapping between service classes and interface queues, and service class policies.

- All service class policies in the destination plan file, as well as all mappings between service classes and interface queues, are deleted.
- Service classes that are assigned to demands in the destination plan and that are not contained in the source plan remain, but their mappings and policies are removed.



- 
- Step 1** Choose **File > Import > QoS Model**.
- Step 2** Browse to or enter the full path of the plan file from which you are importing the QoS model, and click **OK**.
- 

For More Information...	See...
QoS modeling, including service classes, service class policies, and interface queues	<i>Cisco WAE Design User Guide</i>
import_qos CLI tool	import_qos Help Output

## Import Demand Groupings

Demand groupings define a group of demands and provide a convenient way of specifying aggregated traffic in a plan, which can be used as a basis for growth forecasting.

When importing demand groupings, you have the option of importing demand groupings from either another plan file or from a file containing a <DemandGroupings> table. You can combine the imported demand groupings with existing demand groupings as follows.

- **Delete**—Delete all existing demand groupings in the destination plan file. Copy all demand groupings from the source file.
- **Keep**—Do not delete demand groupings in the destination plan file. Copy only demand groupings from the source file that have different names than those in the destination plan.
- **Update**—Do not delete demand groupings in the destination plan file. Copy all demand groupings from the source file. If the two plans have demand groupings of the same name, the demand grouping in the destination plan is replaced.

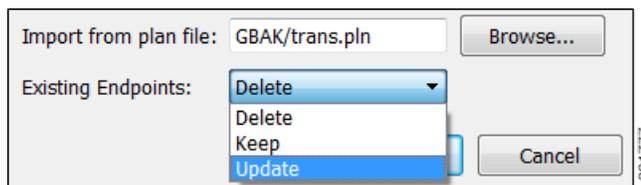
- 
- Step 1** Choose **File > Import > Demand Groupings**.
- Step 2** Select one of these import methods.
- Browse to or enter the full path of the plan file from which you are importing a demand grouping.
  - Browse to or enter the full path of the file containing the <DemandGroupings> table you are importing.
- Step 3** Select how you want the existing demand groupings handled (Delete, Keep, Update).
- Step 4** Select whether to import tags for sites, nodes, AS's, and demands, and click **OK**.
-

For More Information...	See...
<ul style="list-style-type: none"> <li>• Demands</li> <li>• Demand Groupings</li> </ul>	<i>Cisco WAE Design User Guide</i>
Create demand groupings	<i>Cisco WAE Design User Guide</i> insert_demand_grouping_mesh Help output
Demand grouping traffic	<i>Cisco WAE Design User Guide</i> import_demand_grouping_mesh Help output
Use demand groupings in growth plans	<i>Cisco WAE Design User Guide</i> create_growth_plans Help output

## Import External Endpoints

When importing external endpoints, you can combine them with existing external endpoints as follows.

- **Delete**—Delete all existing external endpoints in the destination plan file. Copy all external endpoints from the source plan file.
- **Keep**—Do not delete external endpoints in the destination plan file. Copy only external endpoints from the source plan file that do not have the same name as external endpoints existing in the destination plan.
- **Update**—Do not delete external endpoints in the destination plan file. Copy all external endpoints from the source plan file. If the two plans have external endpoints of the same name, the external endpoint in the destination plan is replaced.



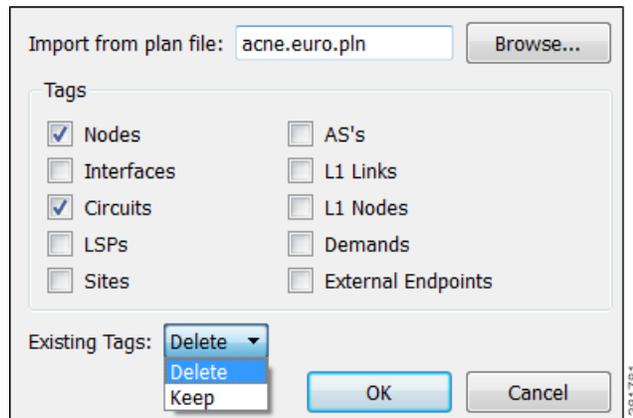
- 
- Step 1** Choose **File > Import > External Endpoints**.
- Step 2** Browse to or enter the full path of the plan file from which you are importing external endpoints.
- Step 3** Select how you want the existing external endpoints handled (Delete, Keep, Update), and click **OK**.
- 

For More Information...	See...
External endpoints	<i>Cisco WAE Design User Guide</i>
import_external_endpoints CLI tool	import_external_endpoints Help output

# Import Tags

When importing tags, you can keep or delete existing tags as follows.

- **Delete**—Delete all existing tags in the destination plan file. Copy all tags from the source plan file.
- **Keep**—Do not delete tags in the destination plan file. Copy only tags from the source plan file that do not have the same name as tags existing in the destination plan.



- 
- Step 1** Choose **File > Import > Tags**.
- Step 2** Browse to or enter the full path of the plan file from which you are importing tags.
- Step 3** Select one or more object types for which you are importing tags.
- Step 4** Select how you want the existing tags handled (Delete, Keep), and click **OK**.
- 

For More Information...	See...
Tags	<i>Cisco WAE Design User Guide</i>
import_tags CLI tool	import_tags Help output





## Importing Traffic and Growth Rates

---

This chapter describes how to import traffic and traffic growth rates into a plan file. It also describes how demand grouping traffic can be used in forecasting, as well as *representative* plan files, which combine traffic from multiple files.

### Traffic and Growth Rate Imports

You can import information in plan files:

- Traffic for existing interfaces, nodes, LSPs, ports, demands, and flows.
- New demands and flows with traffic.
- Growth rates for interfaces and demands. You have the option to import growth rates per interface, per demand, or per tag.

You can import traffic from one plan file into another. You can also import traffic from other data sources using files containing tab-delimited traffic tables.

### Import Rules

WAE Design applies the following rules when importing traffic and growth. See [Table 4-1](#) for a list of required tables, depending on what you are importing.

- Regardless of the object you are importing and regardless of whether importing traffic or growth, a traffic table must exist either within a plan file or within tab-delimited tables. When you import the traffic, the traffic tables are imported directly into the plan file.
- The interfaces, nodes, LSPs, and ports being imported must exist in the plan file into which you are importing traffic. For example, if importing interfaces, the Interfaces (<Interfaces>) table must exist and each interface identified in the traffic table must exist within the Interfaces table. If an object is in the traffic table, but is not in the plan file into which you are importing the traffic, the object is ignored.

The same is true for demands and flows if they are not new. If they are new and do not yet exist in the plan file, you must create an object table (Demands or Flows table) for them in addition to creating their traffic table.

- Interface and demand growth rates imported using tags require an additional import growth table. This table is not directly imported into the plan, but rather is applied to the GrowthPercent column in the traffic table. The growth percent is then applied to all the interfaces or demands that have a tag corresponding to the Tag column in the import growth table.

**Table 4-1** Required Tables for Importing Traffic and Growth Rates

If Importing...	Use These Tables...	See...
Interface traffic	<InterfaceTraffic>	<a href="#">Table 4-2</a>
Growth rates per interface	<InterfaceTraffic>	<a href="#">Table 4-2</a>
Growth rates per interface tag	<InterfaceTraffic> and <InterfacesGrowth>	<a href="#">Table 4-2</a> and <a href="#">Table 4-8</a>
Node traffic	<NodeTraffic>	<a href="#">Table 4-3</a>
LSP traffic	<LSPTraffic>	<a href="#">Table 4-4</a>
Ports traffic	<PortTraffic>	<a href="#">Table 4-5</a>
Demand traffic for existing demands	<DemandTraffic>	<a href="#">Table 4-6</a>
Demand traffic for new demands	<DemandTraffic> and <Demands>	<a href="#">Table 4-6</a>
Growth rates per demand	<DemandTraffic>	<a href="#">Table 4-6</a>
Growth rates per demand tag	<DemandTraffic> and <DemandsGrowth>	<a href="#">Table 4-6</a> and <a href="#">Table 4-8</a>
Flow traffic for existing flows	<FlowTraffic>	<a href="#">Table 4-7</a>
Flow traffic for new flows	<FlowTraffic> and <Flows>	<a href="#">Table 4-7</a>

## Traffic Tables

To import traffic, you must have a viable traffic table for each type of traffic you are importing ([Table 4-1](#)). Possible sources for these tables are:

- An existing plan file.
- A .txt file containing the traffic table, which might be manually created or derived from other sources.

Each traffic table must be tab-delimited and contain required “key” columns (see [Table 4-2](#) through [Table 4-7](#)). You can modify the tables using Excel or a text editor.

**Table 4-2** <InterfaceTraffic> Table

Column	Description
Node	You must either specify both the name of the source node (Node) and the interface name (Interface), or you must specify the IP address (IPAddress). The interface must exist in the <Interfaces> table using either the same node and interface combination or using the same IP address.
Interface	
IPAddress	
Queue	Use if you are importing traffic into a specific interface queue. If it does not exist, the interface queue will be created. If omitted, traffic is imported as undifferentiated.
TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
TraffMeas	Enter the amount of traffic (in Mbps) that you are importing. Required if importing traffic.
GrowthPercent	Enter the percentage by which you are growing the traffic. Required if importing growth rates.

**Table 4-3** <NodeTraffic> Table

Column	Description
Node IP Address	You must either specify the node name (Node) or the IP address (IPAddress). The node must exist in the <Nodes> table using either the same node name or IP address.
Queue	Use if you are importing traffic into a specific interface queue. If it does not exist, the interface queue will be created. If omitted, traffic is imported as undifferentiated.
TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
SrcTraffMeas	Amount of in Mbps that is leaving the node.
DestTraffMeas	Amount of traffic in Mbps that is destined for the node.

**Table 4-4** <LSPTraffic> Table

Column	Description
Name Source SourceIP	You must either specify both the LSP name (Name) and source node name (Source), or you must specify the IP address (SourceIP). The LSP must exist in the <LSPs> table using either the same LSP name and source node name combination or using the same IP address for the source node.
Queue	Use if you are importing traffic into a specific interface queue. If it does not exist, the interface queue will be created. If omitted, traffic is imported as undifferentiated.
TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
TraffMeas	Enter the amount of traffic (in Mbps) that you are importing. Required if importing traffic.

**Table 4-5** <PortTraffic> Table

Column	Description
Node	Name of the node containing the port. Must reside in the <Nodes> table.
Port	Port name. Must reside in the <Ports> table with the same node that is identified in the Node column.
Queue	Use if you are importing traffic into a specific interface queue. If it does not exist, the interface queue will be created. If omitted, traffic is imported as undifferentiated.
TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
TraffMeas	Enter the amount of traffic (in Mbps) that you are importing. Required if importing traffic.

**Table 4-6** Required Columns for <Demands> and <DemandTraffic> Tables

Table	Column	Description
<Demands> and <DemandTraffic>	Name	Demand name. If not specified, the default is empty.
	Source	Name of the source node. Must exist in the <Nodes> table.
	Destination	Name of the destination node. Must exist in the <Nodes> table.
	ServiceClass	Use if you are importing traffic into a specific service class. If it does not exist, the service class will be created. If omitted, traffic is imported to the Default service class.
<DemandTraffic>	TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
	Traffic	Enter the amount of traffic (in Mbps) that you are importing. Required if importing traffic.
	GrowthPercent	Enter the percentage by which you are growing the demand traffic. Required if importing growth rates.

**Table 4-7** Required Columns for <Flows> and <FlowTraffic> Tables

Table	Column	Description
<Flows> and <FlowTraffic>	FlowID	Demand name. If not specified, the default is empty.
<FlowTraffic>	TrafficLevel	Use if you are importing traffic into a specific travel level. If it does not exist, the traffic level will be created. If omitted, traffic is imported to the Default traffic level.
	Queue	Use if you are importing traffic into a specific interface queue. If it does not exist, the interface queue will be created. If omitted, traffic is imported as undifferentiated.
	TrafficMeas	Enter the amount of traffic (in Mbps) that you are importing.

## Growth Tables

Using a growth table, you can apply growth rates to only tagged interfaces or tagged demands. The parameters identified in a growth table are applied to an associated traffic table, but are not directly imported as a table into a plan file.

There are two growth tables: <InterfacesGrowth> and <DemandsGrowth>. Each of these have only two columns.

**Table 4-8** Columns in <InterfacesGrowth> and <DemandsGrowth> Tables

Column	Description
GrowthPercent	The growth rate for the undifferentiated service class in one time period. This GrowthPercent overwrites the GrowthPercent column in the <InterfaceTraffic> or <DemandTraffic> table, depending on what you are importing.
Tag	A tag value for the growth rate, which associates it with interfaces or demands that have the same tag value. Therefore, a Tag column with this tag value must exist in the <Interfaces> or <Demands> table, depending on what you are importing.

This is an efficient means of simultaneously applying growth rates to numerous interfaces or demands at one time. For example, if you have a 1,000 demands that used two tags, you could create a <DemandsGrowth> table with only two rows (one for each tag) versus creating a <DemandTraffic> table with 1,000 entries (one per demand).

## Import Traffic and Growth Rates

To Import...	Use...	Table Prerequisites
Traffic into interfaces, nodes, ports, demands, or flows	File > Import > Traffic in GUI or <code>import_traffic</code> CLI tool	Object tables <a href="#">Traffic Tables</a>
Traffic into LSPs	File > Import > Traffic in GUI or <code>import_lsp</code> CLI tool	LSP (object) table <a href="#">Traffic Tables</a>
Growth rates for interfaces or demands on per-object basis	File > Import > Traffic in GUI or <code>import_traffic</code> CLI tool	Object tables <a href="#">Traffic Tables</a>
Growth rates for interfaces or demands based on tags	File > Import > Traffic in GUI or <code>import_growth</code> CLI tool	Object tables <a href="#">Traffic Tables</a> <a href="#">Growth Tables</a>

### Import Traffic and Per-Object Growth Rates Using the GUI

- 
- Step 1** Choose **File > Import > Traffic**. The Import Traffic dialog box opens.
- Step 2** Import traffic measurements.
- a. In the Traffic File area, select whether to import from a plan or from a table file, and then enter or browse to the plan or table filename you are importing.
  - b. In the Traffic to Import area, select the traffic tables you want to import: interfaces, LSPs, nodes, ports, flows, or demands.
    - If importing interfaces, LSPs, or nodes from a table, specify the columns to use as the table key. For example, for interfaces, select whether to identify the traffic imported based on the IP address or based on a combination of the node and interface name.
    - For flows and demands, the “new” option imports flows or demands that do not exist in the plan file, including their traffic. The “in plan” option imports traffic only for flows or demands that currently exist in the plan.
- Step 3** In the Growth Rates to Import area, select whether to import growth rates for interfaces and demands. This import is on a per-object basis. To import traffic on a per-tag basis, see [Import Growth Rates Based on Tags Using the GUI](#).
- Step 4** In the Traffic Levels area, select whether to import all the traffic levels or a single traffic level. If importing all, the traffic level names are kept on import. If importing a single traffic level when there are multiple levels available, you must specify a name. Optionally, the level can be renamed on import.
- Step 5** To display a plan with the imported information, click **Preview** and then **OK**.
- Step 6** Click **OK**.
-

You can also import demands by creating a demand mesh table and then import it while creating a demand mesh. This is specified by the `-demandmesh-table` option of the `dmd_mesh_creator` tool. The `DemandMeshGenerator` table provides information on the demand mesh to create. It has the following columns:

- **Endpoint**—Represents the source/destination of a demand. For nodes, this corresponds to the node name. Other types of endpoints can also be specified. For example, 'AS{ 12345}' is a valid entry for an AS.
- **SrcDest**—Has a value of *Src*, *Dest*, or *both*, which specifies whether to use the endpoint as the source of the new demands, as the destination, or as both.
- **SrcTraffic**—Represents the source traffic (used by the `dmd_traffic_creator` tool).
- **DestTraffic**—Represents the destination traffic (used by the `dmd_traffic_creator` tool).

## Import Growth Rates Based on Tags Using the GUI

- 
- Step 1** Choose **File > Import > Growth**. The Import Growth dialog box opens.
- Step 2** Select whether to import demand or interface growth rates.
- Step 3** If the interfaces or demands into which you are importing traffic have multiple tags, specify how to apply them.
- Average the growth rate evenly across the tags
  - Add growth rate to each tag
- Example: An interface has two tags: `GrowthTrend` and `ExtraSales`. Each tag is listed in the `<InterfacesGrowth>` table, but `Peak` has a growth of 10%, whereas `Normal` has a growth of 5%.
- If averaged, the resulting growth rate is 7.5%.
  - If added, the resulting growth rate is 15%.
- Step 4** Click **Preview** to see how many rows in the growth table apply to the interfaces or demands.
- Step 5** Click **OK**.
- 

## Demand Grouping Traffic

WAE Design lets you specify traffic growth per demand grouping, which in turn lets you create growth plans that use estimates of aggregate traffic growth. Demand grouping traffic can be entered into WAE Design in several ways:

- Enter the traffic growth into the Demand Grouping Properties dialog box in the WAE Design GUI.
- Import a single period of demand grouping traffic using the Import Demand Grouping Traffic tool.
- Create an external `<DemandGroupingTraffic>` table and select the option to use this table in the Create Growth Plan GUI tool or `create_growth_plans` CLI tool.

## <DemandGroupingTraffic> Table

The <DemandGroupingTraffic> table specifies traffic growth for demand groupings, over one or more periods. You can then import the traffic growth values for a single period or use the entire table to create growth plans for multiple periods.

For More Information...	See...
Create growth plans for multiple periods using the <DemandGroupingTraffic> table	<i>Cisco WAE Design User Guide</i> create_growth_plans Help output
Import single periods of the <DemandGroupingTraffic> table for use in creating growth plans	<a href="#">Import Demand Grouping Traffic Growth</a> insert_demand_grouping Help output

### Table Content

Each row in the <DemandGroupingTraffic> table specifies forecasts for a demand grouping that must already exist in the plan file into which this table is being imported. (These demand groupings must exist in a table called <DemandGroupings>.)

WAE Design calculates the expected traffic based on a combination of columns and periods where the <period> variable specifies the time period. [Table 4-9](#) describes the columns; the examples are for the simplest forecast calculation possible where only one column and one period are used.

**Table 4-9** <DemandGroupingTraffic> Columns

Column Name	Description
DemandGrouping	Name of the demand grouping. This name must exist in the <DemandGroupings> in the plan file into which you are importing the table.
TrafficLevel	Specifies that the growth is to be applied to this traffic level, which must exist in the plan file into which the table is being imported. If not specified, the default is to apply the growth to all traffic levels.
TrafficTotal:<period>	Specifies an exact traffic total in Mbps. If not specified, use the total traffic for the demand grouping of the previous period, or the base plan if this is the first period.
GrowthPercent:<period>	Specifies the percent by which to grow current traffic in Mbps. Example: If traffic total is 40,000 and GrowthPercent is 10, the growth plan grows the traffic to 44,000 Mbps.
TrafficIncrement:<period>	Specifies the amount of traffic to increment by in Mbps. Example: If traffic total is 2600 and TrafficIncrement is 500, the growth plan grows the traffic to 3100 Mbps. Note: If both GrowthPercent and TrafficIncrement are used for the same <period>, the GrowthPercent is applied first.

### Traffic Growth Columns and Periods

The TrafficTotal, TrafficIncrement, and GrowthPercent columns let you generate forecasts for multiple time periods, and each column defines a different manner in which traffic will be increased. Each column uses a <period> variable that serves several purposes. See [Example <DemandGroupingTraffic> Table](#).

- The order in which <period> variables appear in the table defines the order in which WAE Design defines the growth plans.
- The <period> names the period being forecasted, for example Q1, and this name is appended to the root filename when new growth plan files are created. (Root filename is the plan file into which you are importing the table.)
- The <period> tells WAE Design which of the columns to combine to calculate the forecast for that one period.

If multiple columns use the same <period> name, WAE Design uses all of those columns to generate one forecast for that <period>. For example, if both GrowthPercent:Q1 and TrafficIncrement:Q1 are defined, then both columns are used in the calculation.

Columns can be (and likely are) repeated, each time with a different <period> variable. For example, if GrowthPercent:Q1 and GrowthPercent:Q2 are both defined, they are used for the creation of two separate plan files.

## Traffic Growth Calculations

It is the combination of the columns and their use of <period> that WAE Design uses to generate growth plans, or expected traffic, for the sum of all demands within each demand grouping. For each period associated with a demand grouping the growth plan is determined using the following algorithm. See [Example <DemandGroupingTraffic> Table](#).

- 
- Step 1** If TrafficTotal is specified, the expected traffic is set to this value.
- If TrafficTotal is not specified and if this is the first period listed in table, the expected traffic is the total traffic in the demand grouping in plan file into which it is being imported (*base plan*).
- If TrafficTotal is not specified and if this is not the first period, the expected traffic is the total traffic in the demand grouping for the previous period.
- Step 2** If GrowthPercent is specified, the traffic growth is according to this percentage.
- Step 3** If TrafficIncrement is specified, this value is added to the expected traffic after GrowthPercent is applied.
- 

## Example <DemandGroupingTraffic> Table

[Table 4-10](#) is an example <DemandGroupingTraffic> table. Each row identifies a unique demand grouping in the ACME plan file, together with forecast data for future periods.

- TYO to All demand grouping is forecasted to grow by 1,000 Mbps in Q1 and an additional 5% in Q2.
- SEL to All demand grouping is forecasted to grow by 1,500 Mbps in Q1. In Q2, it is forecasted to grow by 5%, plus an additional 2,000 Mbps.
- PEK to All demand grouping is forecasted to have 5,000 Mbps of traffic in Q1 and grow 6% in Q2.

**Table 4-10** Example <DemandGroupingTraffic> Table

DemandGrouping	TrafficTotal:Q1	TrafficIncrement:Q1	GrowthPercent:Q2	TrafficIncrement:Q2
TYO to All		1000	5	
SEL to All		1500	5	2000
PEK to All	5000		6	

Table 4-11 identifies the results of applying this example <DemandGroupingTraffic> table to the ACME plan file. The newly created plan files are named ACME\_Q1 and ACME\_Q2.

- The Current Traffic column lists the total traffic across all demands in the associated demand grouping. This demand grouping resides in the plan file into which this table is being imported.
- The Q1 Expected Traffic column lists the traffic forecasted for that Q1 period using the rules specified in [Traffic Growth Calculations](#).  
Example: SEL to All demand grouping is 1,500 Mbps increment + 3,000 Mbps of existing traffic = 4,500 Mbps.
- The Q2 Expected Traffic column lists the traffic forecasted using these same rules, only this time using Q1 traffic as the basis for calculations.  
Example: SEL to All demand grouping is (5% growth of 4,500) + 2,000 Mbps increment = 6,725 Mbps.

**Table 4-11** Expected Q1 and Q2 Traffic Growth

Demand Grouping	Current Traffic	Q1 Traffic	Q2 Traffic
TYO to All	2000	3000	3150
SEL to All	3000	4500	6725
PEK to All	4000	5000	5300

## Import Demand Grouping Traffic Growth

You can import demand grouping traffic growth information.

- The demand groupings in the current plan must match the names of the demand groups in the <DemandGroupingTraffic> table.
- The file from which you are importing must be a .txt file.

- 
- Step 1** Choose **File > Import > Demand Grouping Growth**.
- Step 2** In the **Import from table file** field, enter or browse to the file containing the <DemandGroupingTraffic> table.
- Step 3** To import only traffic for a specific period, enter that Period name and then click **OK**. Alternatively, leave the **Period** field empty, and when you click **OK** you are prompted with a list of periods from which to choose.
- 

## Representative Plan Files

Plan files are snapshots of a network state at a point in time. Transitory events, such as failures, are captured in the snapshot if they occur during this time. The traffic collected is the specific traffic that occurred during that collection window, which can be five minutes or less. As such, a snapshot usually does not represent the network state over the course of a typical day or week of network operation, and thus, is inadequate to use as the basis for long-term design and planning tasks. To address these needs, the `create_representative_plan` tool uses multiple snapshots from an archive to construct a single plan that is more representative of the general network state.

- The topology is extracted from a base plan, which by default is the most recent snapshot. You can, however, specify any plan file as the base plan.
- The representative plan contains multiple traffic levels, one per time interval specified. For example, there could be one traffic level per hour of the day.
- Demands are extracted from snapshots that are selected from each of these time intervals. A single demand in the representative plan file contains a range of traffic values representative of the different amounts of traffic for that demand over the course of the specified time period.
- Interface, LSP, and node measurements are extracted from snapshots that are selected from each of these time intervals.

A representative plan file is particularly useful when planning networks where peak utilizations occur at unknown or different times of the day across different interfaces. A simulation analysis performed over all traffic levels identifies the time intervals when peaks occur.

You can fine-tune the results to include specified snapshots for multiple traffic levels across specific intervals. These time parameters are divided into two sets.

- One set includes `-time-period`, `-time-interval-length`, and `-time-interval-starts`. These parameters define time intervals during a day or during a week (defined by `-time-period`) that are created for the resulting plan file.
- The other set includes `-sample-time-end`, `-sample-time-length`, and `-time-zone`. These parameters define which periods of data in the archive are used to populate the specified time intervals.

The `create_representative_plan` tool does the following to create the representative plan file:

- Examines all snapshots that fall into the specified time interval during the sample time range. Of these snapshots, WAE Design selects the one with the least number of failed circuits and the most number of active interfaces in the common base plan. If there is a tie, the snapshot with the highest amount of demand traffic is selected. Only snapshots with single traffic levels are used.
- Removes all traffic intervals from the base plan.
- Creates new traffic intervals in the base plan, using the format `HHMM-HHMM` or `DDDHHMM-DDDHHMM`, depending on whether the time period is a day or a week.  
Examples: `03:00-04:00` and `Fri17:00-Fri18:00`
- Imports all demands from the corresponding snapshot for the time interval.
  - If a snapshot demand matches one existing in the base plan, then WAE Design uses that demand and the snapshot demand traffic for it.
  - If there is no matching demand, WAE Design creates the demand with 0 traffic.
  - If a demands exists in the base plan, but not in the snapshot, the demand is used with 0 traffic.

Note that multicast demands are imported with the required multicast flows and multicast destinations.

- Imports measured traffic for interfaces, LSPs, and nodes that exist in both the base plan and the snapshot.

Each representative plan file includes a report section where each traffic level is defined per row.

## Examples

In this first example, the network peak time is between 4 PM and 7 PM daily. To better understand this peak traffic, we could create a representative traffic level for each hour of this range, which includes 4 PM, 5 PM, and 6 PM. For weekly forecast purposes, we are interested in sampling the last 5 days to construct the traffic levels. We are naming the output file “representative\_day.pln.” We are choosing to use the latest snapshot in the time period, 110502\_0347.UTC.pln, as the base plan, and it is located in the archive that is named “backbone.” See the following command and sample output:

```
create_representative_plan -out-file representative_day.pln -archive backbone
-time-interval-length 60 -sample-time-length 1 -time-interval-starts 1600,1700,1800
```

Traffic Level	Snapshot	Matching Interfaces	Total Demand Traffic	Total Demands	Demands Not Imported
16:00-17:00	110502_0347.UTC.pln	25	43534.32	453	4
17:00-18:00	110502_0347.UTC.pln	25	47583.23	454	3
18:00-19:00	110502_0347.UTC.pln	25	50771.49	454	3

In this second example, we know the network peak times are around 4 PM daily, as well as 8 PM on Fridays. We need to get a representative traffic level for each of these six periods for the last two weeks. Today is Tuesday, so yesterday’s 4-5 PM range and last week’s Monday 4-5 PM range are used to construct the 4 PM traffic level. The name of the representative plan file we are creating is “weekly\_peak.pln.” The base plan, 110407\_0423.UTC.pln, is in the acme directory. See the following command and sample Traffic Levels output:

```
create_representative_plan -plan-dir acme -base-plan 110407_0423.UTC.pln -outfile
weekly_peak.pln -time-period week -time-interval-length 60 -sample-time-length 14
-time-interval-starts Mon1600,Tue1600,Wed1600,Thu1600,Fri1600,Fri2000
```

Traffic Level	Snapshot	Matching Interfaces	Total Demand Traffic	Total Demands	Demands Not Imported
Mon16:00-Mon17:00	110407_0423.UTC.pln	97	53245.14	6702	21
Tue16:00-Tue17:00	110407_0423.UTC.pln	97	53413.36	6702	21
Wed16:00-Wed17:00	110407_0423.UTC.pln	95	49985.27	6701	22
Thu16:00-Thu17:00	110407_0423.UTC.pln	97	56831.91	6702	21
Fri16:00-Fri17:00	110407_0423.UTC.pln	93	48732.18	6700	23
Fri120:00-Fri21:00	110407_0423.UTC.pln	97	53692.39	6702	21





## Exporting Routes

---

The ability to export routes and tables facilitates the exchange of information in a text-editable format. It also supplements the analysis of routes and plan file objects. For example, you can export demand routes using L1 links to determine which L1 links are used by demands. Because you can export routes that changed due to failure states, you can fine-tune your analysis of how failures impact the network.

### Export Routes

You can export routes using the WAE Design GUI or using `export_routes` CLI tool.

This tool creates one of the following tables, depending on what you are exporting:

- <CircuitHops>
- <DemandHops>
- <L1CircuitHops>
- <L1CircuitPathHops>
- <LSPHops>
- <LSPPathHops>
- <ShortestIGPPathHops>
- <ShortestLatencyPathHops>
- <ShortestTEPathHops>

The table generated is output to a .txt file containing only this table. Their key columns are a combination of the key column identified for their object type plus a Step column. The Step column is an integer that identifies the sequence of this hop in the route path. For example, the <Demands> table has key columns of Name, Source, Destination, and ServiceClass. Therefore, the <DemandHops> table has these four columns to uniquely identify the demand, plus a Step column to identify the hop sequence.

Each exported hops table includes the following columns to specify details of the hop and its position with respect to other hops in the route. Note that the routes of demands might be split to account for ECMP routing or multicast. So demand hops might not be listed sequentially as a hop could be followed by two next hops as the route and traffic split into two.

- **UnresolvedHop**—The hop type and complete definition. For example, if `{cr1.atl|to_cr1.hst}` means this hop is an interface with a source node of `cr1.atl` and an egress interface of `to_cr1.hst`. For a complete list of these object notations, see [Plan Files and Tables](#).

- **PreviousStep**—Identifies the previous step or steps in the route. As such, this identifies the previous hop in the route. For example, if you are on the row corresponding to Step 4, and the PreviousStep is 2, then the previous hop is contained in the row corresponding to Step 2.

If the PreviousStep column is empty, then this is the first hop in the route.

- **NextStep**—Identifies the next step in that table. As such, this identifies the next hop in the route.

If the NextStep column is empty, then this is the last hop in the route.

- **TrafficProportion**—The proportion of traffic on this hop compared to the other hops in the route. Although this value is usually 1, in the case of ECMP routing, it could be less than 1.

Optionally, you can fail objects prior to exporting their routes and then export only the routes that changed due to that failed state.

For More Information...	See...
Export routes through WAE Design GUI	<a href="#">Export Routes Using the GUI</a>
Export routes using CLI tool	export_routes Help output
Key column definitions per table	\$CARIDEN_HOME/docs/table_schema.html file; key columns are highlighted in orange
Key columns	<a href="#">Plan Files and Tables</a>
WAE Design tables	<a href="#">Plan Files and Tables</a>
Object notation	<a href="#">Plan Files and Tables</a>
Fail objects prior to exporting routes	<i>Cisco WAE Design User Guide</i>

## Example Export Route Tables

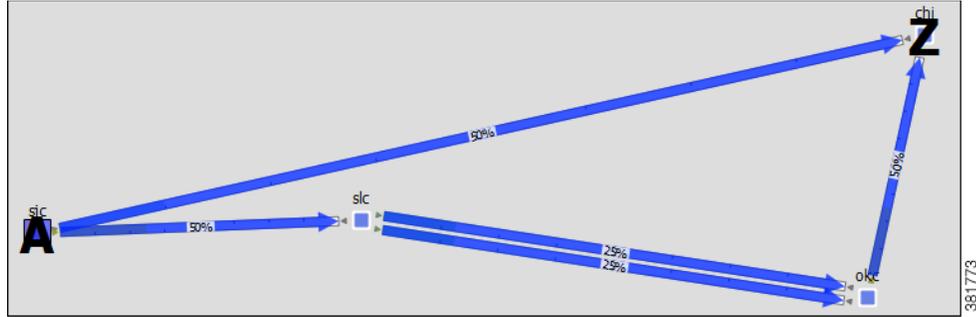
[Table 5-1](#) is an example <LSPHops> table that was created using active LSPs with interface hops. The cr2.par\_cr1.fra LSP has no interface hops in its route. In the table, it has only one step and no previous or next steps. The cr1.ams\_cr1.par LSP has three interface hops (Steps 1, 2, and 3) between source (cr1.ams) and destination (cr1.par).

**Table 5-1** Example <LSPHops> Table

Name	Source	Destination	Step	UnresolvedHop	Previous Step	Next Step	Traffic Proportion
cr2.par_cr1.fra	cr2.par	cr1.fra	1	if{cr2.par to_cr1.fra}			1
cr1.ams_cr1.par	cr1.ams	cr1.par	1	if{cr1.ams to_cr2.lon}		2	1
cr1.ams_cr1.par	cr1.ams	cr1.par	2	if{cr2.lon to_cr1.lon}	1	3	1
cr1.ams_cr1.par	cr1.ams	cr1.par	3	if{cr1.lon to_cr1.par}	2		1

[Table 5-2](#) is an example <DemandHops> table that was created using demands with interface hops ([Figure 5-1](#)). The demand route in this example splits into multiple ECMP paths.

Figure 5-1 Example Demand Route Shown in &lt;DemandHops&gt; Table

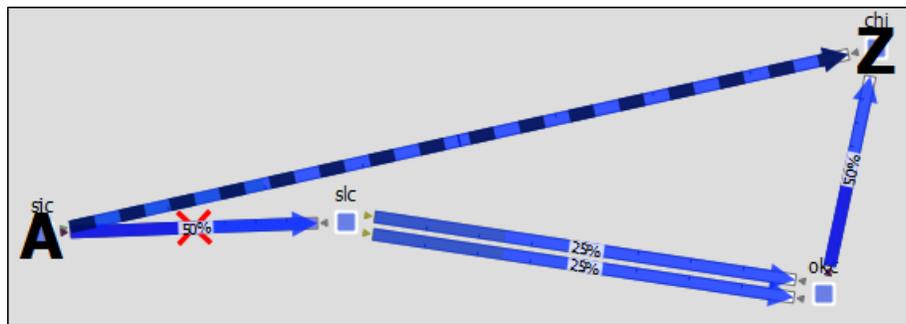


381773

Table 5-2 Example &lt;DemandHops&gt; Table with ECMP Route

Name	Source	Destination	Service Class	Step	UnresolvedHop	Previous Step	Next Step	Traffic Proportion
	sjc	chi	Default	1	if{sjcl{to_slc}}		3;4	0.5
	sjc	chi	Default	2	if{sjcl{to_chi}}			0.5
	sjc	chi	Default	3	if{slcl{to_okc}}	1	5	0.25
	sjc	chi	Default	4	if{slcl{to_okc}[1]}	1	5	0.25
	sjc	chi	Default	5	if{okcl{to_chi}}	3;4		0.5

If you were to fail the circuit between sjc and slc and export only routes that changed during a failed state, the resulting exported route table would be as shown in Table 5-3.



381774

Table 5-3 Example &lt;DemandHops&gt; Table After Failing a Circuit

Name	Source	Destination	Service Class	Step	UnresolvedHop	Previous Step	Next Step	Traffic Proportion
	sjc	chi	Default	1	if{sjcl{to_slc}}			1

## Export Routes Using the GUI

From the WAE Design GUI, follow these steps to export routes to the hops tables. Hops tables are created for the object that is selected in the dialog box. For example, if you select Circuits, WAE Design creates a <CircuitHops> table containing all circuits in the plan file.

If you want to export routes only for those objects whose route changed due failure state, fail all relevant objects before exporting the routes and select the “Only export routes changed by failure state” option. Alternatively, you could run Simulation Analysis and then fail one or more objects to the worst-case failure. For more information, see the *Cisco WAE Design User Guide*.

When exporting circuits, only L3 circuits using L1 hops are exported.

When exporting L3 circuits and L1 circuits, the Active Path Sim property in the L1 Circuits table is considered. This property identifies which L1 circuit path is currently being used by the L1 circuit. If an L1 circuit path is not available due to failure, the L1 circuit’s Active Path Sim is updated to the L1 circuit path that is being used. When exporting L1 circuit paths, all routed paths are exported.

- 
- Step 1** Choose **Export > Export Routes**. The Export Routes dialog box opens.
  - Step 2** Select the object for which you want to export routes.
  - Step 3** Except for circuits and L1 circuits, select the hop type that you want to export. For circuits, L1 circuits, and L1 circuit paths, the only available option is Layer 1 links hops.
  - Step 4** If exporting only routes that changed due to a failure, select the associated option.
  - Step 5** Click **OK**, and save the .txt file to the location of your choice.
- 

## Export Explicit LSP Path Settings

Tab-delimited .txt files listing all explicit LSP path settings are created by choosing **File > Export > Explicit LSP Settings** in the WAE Design GUI, or using `export_routes` CLI tool. The Hops column lists the explicitly routed path of each LSP in object notation. For a complete list of these object notations, see [Plan Files and Tables](#).

[Table 5-4](#) is an example output file. In addition to these columns, the file also contains SetupBW, SetupPri, and HoldPri columns. [Figure 5-2](#) shows the first LSP path option listed in [Table 5-4](#).

**Figure 5-2 Path Option 1 for the cr1.ams\_er1.BRU LSP**

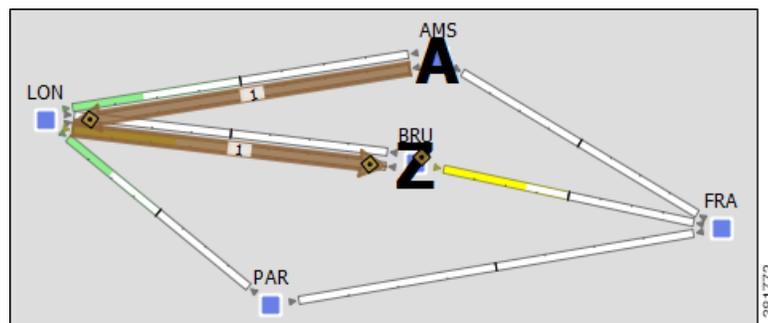


Table 5-4 Example File Containing Explicit LSP Path Settings

Name	LSP	Source	Destination	Preference	Hops
cr1.ams_er1.BRU_1	cr1.ams_er1.BRU	cr1.ams	er1.BRU	1	if{er1.BRU {to_cr1.BRU}},if{cr1.BRU {to_cr2.lon}},if{cr2.lon {to_cr1.ams}}
cr1.ams_er1.BRU_1	cr1.ams_er1.BRU	cr1.ams	er1.BRU	2	if{er1.BRU {to_cr2.BRU}},if{cr2.BRU {to_cr1.lon}},if{cr1.lon {to_cr2.ams}},if{cr2.ams {to_cr1.ams}}

## Export L1 Link Wavelength Utilization

Changes in the routing and Lambdas of L1 circuits paths affect the utilized wavelengths on L1 links. Any L1 failure or a site failure could cause the L1 circuit path to be rerouted. If Auto Lambda is enabled, then the L1 circuit path could be forced to select a Lambda on a different path option in order to avoid lambda blocking. For more information about Layer 1, see the *Cisco WAE Design User Guide*.

The Export Lambda Utilization tool lets you create and export a file containing a <LambdaUtil> table that identifies all of the L1 circuit path Lambda Sim values that are routed on each L1 link. This is useful for determining how many L1 circuits paths are using an L1 link, to determine how fragmented the wavelength usage is across L1 links, and to perform more elaborate wavelength assignments.

The table shows whether there is a failure, the L1 link, and the Lambda values of the L1 circuit paths traversing it. Each Lambda column correlates to a different L1 circuit path. To see the names of the L1 circuit paths, use the “Specify L1 circuit paths” option.

Example: In this simple network there are no failures, and only four Lambdas are used. The ams-lon L1 link has an L1 circuit path using Lambda1 and none using the remaining Lambdas. The lon-par L1 link has four L1 circuit paths using it.

### <LambdaUtil>

Failure	Name	L1NodeA	L1NodeB	Lambda1	Lambda2	Lambda3	Lambda4
none	ams-lon	cr1.ams	cr2.lon	T	F	F	F
none	lon-par	cr2.lon	cr1.par	T	T	T	T

To see the effects of L1 link, L1 node, and sites failures on the wavelength utilization of L1 links, select the relevant failure sets.

Example: Row 2 shows that the L1lnk{romllonlrom-lon} failure causes an L1 circuit path to switch from one Lambda 1 to another lambda on the ams-lon L1 link or causes it to switch path options that it is using. Row 3 shows that the L1lnk{parlromlpar-rom} failure prevents L1 circuit paths from using either Lambda1 or Lambda2 when traversing the ams-lon L1 link.

<LambdaUtil>								
	Failure	Name	L1NodeA	L1NodeB	Lambda1	Lambda2	Lambda3	Lambda4
1	none	ams-lon	cr1.ams	cr2.lon	T	F	F	F
2	L1lnk{romllonlrom-lon}	ams-lon	cr1.ams	cr2.lon	F	T	T	T
3	L1lnk{parlromlpar-rom}	ams-lon	cr1.ams	cr2.lon	F	F	T	T

- 
- Step 1** If you want to export the lambda utilization for only selected L1 links, first select those L1 links.
  - Step 2** Choose **File > Export > Lambda Utilization**.
  - Step 3** Select which L1 links to use: All, those selected, or a selected tag.
  - Step 4** If you want to run Simulation Analysis on L1 links, L1 nodes, or sites, select one or more of these options.
  - Step 5** If you want to know the name of the L1 circuit path for each Lambda referenced in the table, select “Specify L1 circuit paths.” Then click **OK**.
- 

## Export Current Table

To export the table that is currently showing in the WAE Design GUI, choose **File > Export > Table**. Only the columns that are showing appear in the exported table. To show and hide tables and columns, choose **View > Tables**.

You can also export a table or portions of a table using a simple cut-and-paste method. Select the rows and columns, use Ctrl-C (Cmd-C on Mac) to copy the selection, and then paste it to the desired document. If you paste this information into an Excel document, the data is correctly placed into the Excel rows and columns.



## Importing Offline Collections

This chapter describes the WAE Design GUI tools available to discover and retrieve information from router configuration tools in WAE 6.x. These tools are useful for capturing and importing network information into the WAE Design GUI.

- Parse Configs—[Import Router Configuration Files](#)
- Parse IGP—[Import IGP Database](#)

This chapter provides high-level information on accessing the tools from the WAE Design GUI. These same tools are available from the CLI.

### Import Router Configuration Files

The Parse Configs tool reads a set of Cisco, Juniper Networks, or Huawei router configuration files, creates a plan file of the network, and imports it into the WAE Design GUI.

Directory and Files

Create New Plan  
Plan Name: atlantic-gbd.txt

Update Current Plan  
Update nodes that are in:

Either configs or plan  Both configs and plan  Configs, not plan

Data Directory: configs

381429

**Step 1** Choose **File > Get Plan from > Configs**.

**Step 2** Choose whether to create a new plan file or add information to an existing plan.

- For new plans, enter the complete path and plan filename you are creating.
- If updating an existing plan file, choose how to update nodes.
  - Update only if the nodes exist in either config files or a plan file.
  - Update only if the nodes exist in both config files and a plan file.
  - Update only if the nodes exist in config files and do not exist in a plan file.

Objects to Parse

- Base: Interfaces and nodes
- LAG: Link aggregate groups, link-bundle member ports
- SRLG: Shared-risk link groups, link-bundle member ports
- RSVP: RSVP-TE LSPs, LSP paths, and path hops
- MPLS FRR: Fast Reroute LSPs, LSP paths, and path hops
- VPN: Virtual private networks

**Step 3** Enter or browse to the name of the directory containing the router configurations to be parsed.

**Step 4** Choose one or more configuration objects to parse.

**Step 5** Choose whether IGP is OSPF or IS-IS.

- If parsing OSPF, either use all OSPF areas or choose one and enter its area ID as an integer or IP address. The default is area 0.

If multiple process IDs exist, WAE Design uses the first one in the config unless you specify otherwise.

- If parsing multi-level IS-IS, choose whether to use Level 1, Level 2, or both.

If multiple instance IDs exist, WAE Design uses the first one in the config unless you specify otherwise.

IGP

OSPF

OSPF Area:  All  Single Area Area ID:

OSPF Proc ID:

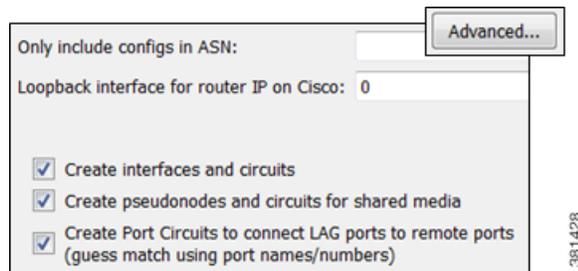
ISIS

ISIS Level:  Level 2  Level 1  Both

ISIS Instance ID:

**Step 6** To set advanced options, click **Advanced**.

- Enter values for these options, as needed, if you prefer not to use the WAE Design defaults.
  - Ignore BGP ASNs.
  - Loopback interface for Cisco routers is 0.
- Choose these options, as needed.
  - Create interfaces and circuits to build a network topology after parsing the configs.
  - Create pseudonodes and interfaces for matching circuits for shared media, such as Ethernet LANs.
  - Create port circuits to connect LAG ports to remote ports. The match between the two is created in ascending order using a combination of port names and port numbers.



**Step 7** Click **OK**.

**Step 8** To save the imported information as a plan file, choose **File > Save as**.

## Import IGP Database

The Parse IGP tool converts IGP information from router `show` commands and imports the IGP database into a new or existing plan file.

**Step 1** Choose **File > Get Plan from > IGP Database**.

**Step 2** Enter or browse to one of the following:

- Directory name containing multiple IGP database files.
- Filename containing the IGP database. This file is the equivalent of the `show` command output for Cisco and Juniper routers.

**Step 3** Choose whether to import an OSPF or IS-IS database.

- For OSPF, choose whether to import OSPFv2 or OSPFv3. Note that OSPFv3 is the protocol that runs in IPv6 networks.

By default, WAE Design collects OSPF area 0 LSDB. To import topologies from non-zero area LSDBs, choose **All**.

- For IS-IS, choose whether to import IS-IS for IPv4 or IPv6. IS-IS topologies must be congruent.

Choose Level 1, Level 2, or both metrics. If you choose the Both option, WAE Design combines both levels into a single network, and Level 2 metrics take precedence.

**Step 4** Choose whether to use DNS to resolve IP addresses (node names) in the resulting plan file. (In the GUI, routers are called *nodes*.)

**Step 5** Click **OK**.

**Step 6** To save the imported information as a plan file, choose **File > Save as**.





## Add-Ons and GUI Customizations

---

WAE Design supports tools for customizing the GUI, enabling you to tailor it to meet your specific needs.

### Add-On Applications

WAE Design provides basic tools for incorporating add-ons (scripts or executables) into the WAE Design GUI.

- CLI tools for editing plan files, such as `table_extract`, `table_edit`, and `table_replace`.
- A framework for creating dialog boxes for data entry to the add-on (see [Create Add-Ons](#)).

Once registered with the WAE Design GUI, add-ons are accessible from the Add-Ons menu. If you do not see an add-on that you expect, choose **Add-Ons > Find Add-Ons**.

### Create Add-Ons

- 
- Step 1** Create an executable to perform the add-on task. This executable processes specific CLI parameters, which the GUI then uses to pass details of plan files, reports, and options specified through the add-on dialog box. See [Construct Add-On Executable](#).
- Step 2** Construct the add-on's configuration file (`addon.txt`), which contains two tables. See [Construct `addon.txt`](#).
- `<AddOnConfigs>`—Defines the add-on name and description, as well as how the executable should be invoked.
  - `<AddOnOptions>`—Defines the WAE Design GUI dialog box from which the executable is invoked from the GUI.
- Step 3** Construct the `return-config-file` parameter of the executable to specify the WAE Design GUI behavior upon exiting the add-on. See [Construct `return-config-file`](#).
- Step 4** Register the add-on with the WAE Design GUI, which is putting the executable and `config.txt` file where they can be discovered by WAE Design. See [Register Add-Ons](#).
-

## Construct Add-On Executable

The add-on can invoke any executable file, such as a shell script or binary executable. The executable must recognize the parameters listed in [Table 7-1](#), which are passed to the add-on executable when it is called from the GUI.

All parameters except `-plan-file` are used by the WAE Design GUI to pass a temporary file or directory name to the add-on script. The script then optionally adds the contents to create the desired file or report. For example, you can create an add-on that generates reports on the existing plan file, as well as one that returns a new plan for display in the GUI.

**Table 7-1** Parameters Passed by the WAE Design GUI to the Add-On Executable

Parameter	Description
plan-file	Required: The plan file in .pln format that is currently active in the GUI.
out-file	Optional: A filename to which the add-on saves a modified version of the plan file. The WAE Design GUI opens this new plan file after the add-on has completed.
pdf-report-file	Optional: A filename to which the add-on writes a report in PDF format. When exiting the add-on, this report is opened using the default PDF viewer. For information on creating PDF reports, see <a href="#">Reporting Tools</a> .
report-dir	Optional: A directory name in which the add-on creates a report that is imported into the currently active plan file and opens the report to its first section. The ShowReport value in the <AddOnReturnConfig> table can show this same report. The ShowReport value also overrides the report-dir value. See <a href="#">Construct return-config-file</a> . For information on creating reports that can be stored in plan files, see <a href="#">Reporting Tools</a> .
report-file	Optional: A filename to which the add-on writes a plain text report section. The WAE Design GUI places this report section in an Add-Ons report in the active plan file. This parameter is available for backward compatibility. We recommend you use the report-dir parameter instead. The ShowReport value in the <AddOnReturnConfig> table overrides this property value. See <a href="#">Construct return-config-file</a> .
return-config-file	The filename to which the add-on writes the instructions on WAE Design GUI behavior after the add-on has completed. This file contains an <AddOnReturnConfig> table. For information, see <a href="#">Construct return-config-file</a> .  This parameter is required if using the report-file parameter.

## Error Messages and Warnings

The add-on can generate error messages and warnings by writing them to the standard output to the GUI log window (accessed via **Window > Log Window**).

- Error messages must start with the word 'Error:' on a new line.
- Warning messages must start with the word 'Warning:' on a new line.

The add-on can inform the WAE Design GUI of its progress by writing the percentage completed to the standard output. This must be a single line containing a number followed by the '%' character. The GUI displays this progress in a progress bar while the add-on is running.

A return value of 0 from the add-on indicates that the add-on ran successfully. Any other value indicates an add-on failure.

## Construct addon.txt

The `addon.txt` configuration file contains two tables.

- `<AddOnConfigs>`—Defines the add-on name, provides a description of the add-on, and identifies how to invoke the executable should be invoked (Table 7-2).
- `<AddOnOptions>`—Optional: Defines input fields in the add-on dialog box (Table 7-3). A CLI parameter is specified for each field, and these are passed to the executable, together with the input values. The rows are sequentially read.

**Table 7-2** `<AddOnConfigs>`

Property	Value
Description	Description displayed in the dialog box.
Name	Name displayed in the dialog box. The name displayed in the Add-Ons menu is a subdirectory name.
PlanFormat	This value determines the type of column information are provided to the add-on. The advantage to using .txt or .db is that you do not have to extract the derived information using other means, such as <code>table_extract</code> .  Default = pln; only plan file column information Options = txt, db; plan file and derived column information
Run	Required: The command used to execute the add-on.
Version	Version number displayed in the dialog box.
WindowsRun	The command used to execute the add-on under Windows, if different than the Run parameter.

**Table 7-3** `<AddOnOptions>` Table

Column	Description
Tab	Create tabs and group options within them. <ul style="list-style-type: none"> <li>• If left empty, the tab is unnamed. If only an unnamed tab is present, it is not displayed and instead the options are shown in a dialog box without tabs.</li> <li>• If the Tab column has the same value as a previously named Tab value, the option is added to the same tab, and is placed below the previous option. If the value is different, a new tab is created and the option is placed in that tab.</li> </ul>
Group	Create groups and place options within them. <ul style="list-style-type: none"> <li>• If there is no Group value, an option is not grouped, and if applicable, is placed below the last ungrouped option.</li> <li>• If the Group value is new, a new group is created within its associated tab. If that row does not have a Tab value, the group is created within an unnamed tab.</li> <li>• If the Group column has the same value as a previously named Group value, the option is added to that same group and is placed below the previous option provided they have the same Tab value.</li> </ul>

Table 7-3 &lt;AddOnOptions&gt; Table (continued)

Column	Description
OptionName	<p>The name of the command-line option; this is prepended with a -(dash) when passed to the executable. The following names are reserved and cannot be used:</p> <ul style="list-style-type: none"> <li>• mate-version</li> <li>• out-file</li> <li>• pdf-report-file</li> <li>• plan-file</li> <li>• report-dir</li> <li>• report-file</li> <li>• return-config-file</li> </ul>
Prompt	Text string that prompts for the desired value.
Type	<p>Type of entry.</p> <ul style="list-style-type: none"> <li>• Bool—Drop-down list with True and False selections. Passes “true” or “false” to the script.</li> <li>• CheckBox—Toggle entry; if selected, it is true, and if not selected, it is false.</li> <li>• ComboBox—Selection of items to appear in a drop-down menu defined by a semicolon-delimited set of options. The user can either select from the menu or type an entry explicitly.</li> <li>• DropDown—Selection of items to appear in a drop-down menu defined by a semicolon-delimited set of options. The user can select from the menu.</li> <li>• Float—Floating point number. Range is expressed as x;y.</li> <li>• Integer—Integer. Range is expressed as x;y.</li> <li>• Password—Text string. Characters typed by the user are hidden.</li> <li>• OpenDir—Directory name. Same as OpenFile but for directory names.</li> <li>• OpenFile—Name of an existing file. Default specifies the filename (no path) in the current directory. Can browse with a browse button to a different directory. The complete path is passed to the script.</li> <li>• SaveFile—Filename. Same as OpenFile, except it can create a file.</li> <li>• String—Text string. Range is ignored.</li> <li>• TableSelection—Drop-down list for the objects in a specified table. For example, if you specify TableSelection in the Type column and Interfaces in the Table column, the drop-down list gives you standard WAE Design choices of selected, all, none, or tags. The Range and Default columns are ignored if this type is specified.</li> <li>• TableEntry—Drop-down list containing the information for traffic levels, service classes, or interface queues. For example, if you have multiple traffic levels in the plan file, this drop-down list lets you choose one of the available traffic levels. Specify the list in the Table column. The selected item is passed to the script as an argument for the option in the OptionName column.</li> </ul>
Table	<p>The table for the TableSelection type or the TableEntry type.</p> <ul style="list-style-type: none"> <li>• TableSelection options can be any table that is accessible from the WAE Design GUI using <b>View &gt; Tables &gt; Show/Hide Tables</b>, except for the Shortest Paths table.</li> <li>• TableEntry is either Queues, ServiceClasses, or TrafficLevels.</li> </ul>

**Table 7-3** <AddOnOptions> Table (continued)

Column	Description
Range	Varies depending on the Type identified.
Default	Varies depending on the Type identified.

## Construct return-config-file

The file named by the `return-config-file` parameter stores the return configuration information in the <AddOnReturnConfig> table (Table 7-4). This table contains a pair of columns, Property and Value, where the values define the WAE Design GUI behavior upon exiting the add-on. For information on the GUI behavior of views, traffic levels, service classes, and queues see the *Cisco WAE Design User Guide*. For information on plot layouts and traffic utilization colors, see the *Cisco WAE Network Visualization Guide*.

**Table 7-4** <AddOnReturnConfig> Properties and Values

Property	Value
OutputFileName	Output (resulting) plan filename. The string can contain a \$1 variable, which is the name of the input plan file (without file extension). The default is \$1-out.  Example: If the input plan filename is euro_geo.pln, by default the output filename is euro_geo-out.pln.
PlotLayout	The GUI displays the specified layout view. If not specified, the GUI displays the Default layout, which is customary for a newly opened plan file.
Queue	The GUI displays traffic for the specified queue. If both ServiceClass and Queue are specified, the ServiceClass value has priority.
ServiceClass	The GUI displays traffic for the specified service class. If not specified, the GUI displays undifferentiated traffic.
ShowReport	The GUI opens the report, including the report-dir report, which should be inserted into the plan before this rule is applied.
ShowReportSection	The report opened by ShowReport opens to this section. If not specified, the report opens to the first section.
TrafficLevel	The GUI displays this traffic level. If not specified, the GUI displays the Default traffic level, which is customary for a newly opened plan file.
UtilColorList	The GUI fills interfaces with these colors to show levels of traffic utilization. If not specified, the default threshold settings and colors are used. Following is the format, where threshold is a real number and colors are of the form #RRGGBB (standard HTML color names):  Format: threshold%<=color<=threshold%  Example: 0% <= red <= 25%, 25% <=green<=50%, 50%<=purple<=75%, 75%<=blue<=100%

Table 7-4 &lt;AddOnReturnConfig&gt; Properties and Values (continued)

Property	Value
UserView	The GUI displays this user-defined plot view. If both View and UserView are specified, the View value has priority. For information on creating plot views, see <a href="#">Customized Network Plot Views</a> .
View	The GUI displays one of these views. If not specified, the GUI displays whichever view is appropriate for the output plan file: <ul style="list-style-type: none"> <li>• FailureImpact—Failure Impact view</li> <li>• MeasUtil—Measured Traffic view</li> <li>• SimLSPRsrv—LSP Reservations view</li> <li>• SimUtil—Simulated Traffic view</li> <li>• SimWCUtil—Worst-Case Traffic view</li> </ul>

## Register Add-Ons

To register an add-on with WAE Design, create a subdirectory for the add-on in the `$CARIDEN_HOME/addons` directory. At a minimum, this directory must contain the `addon.txt` configuration file that defines the dialog box and information about the executable.

WAE Design adds a menu item in the Add-Ons menu using the Name parameter in the <AddOnConfigs> table. If this parameter is omitted, WAE Design uses the name of the subdirectory.

To create multiple levels of submenus in the WAE Design GUI, simply create nested subdirectories. See [Figure 7-1](#) for an example.

Add-ons and their submenus appear in the Add-Ons menu the next time you launch the GUI, or after you choose **Add-Ons > Find Add-Ons**.

**Figure 7-1 Example Add-On Submenus**

### Directory Structure

```
$CARIDEN_HOME/addons
addons/Reports/highutilcircuits
addons/Reports/trafficdistribution
```

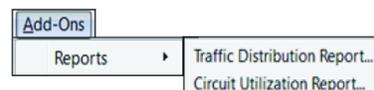
### Names

```
addon.txt <AddOnConfigs>

<AddOnConfigs>
Property Value
Name Circuit Utilization Report

<AddOnConfigs>
Property Value
Name Traffic Distribution Report
```

### Result



381768

## Example Add-Ons

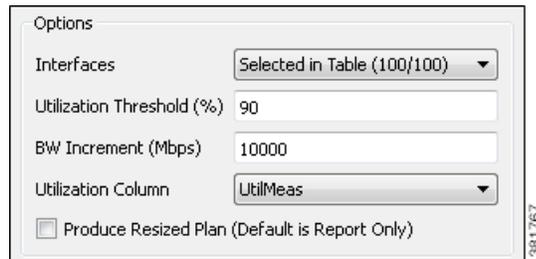
WAE Design includes three example add-ons: Circuit Upgrade, Circuit Utilization Report, and Traffic Distribution Report.

### Circuit Upgrade Add-On

The `$CARIDEN_HOME/addons/CircuitUpgrade` directory includes a sample add-on named `CircuitUpgrade` that analyzes the Interfaces table Utilization columns. Upon exiting the add-on dialog box (Figure 7-2), the add-on executable does the following:

- For every interface whose utilization (in the specified utilization column) exceeds a threshold, the circuit capacity is increased by the specified increment until the condition is met.
- A report named *Circuit Upgrade* is generated. The report contains only one section, which is named *List of Upgrades*.
- The WAE Design GUI opens a new plan file using the same name with *-upgrade* added as a suffix.

**Figure 7-2 Example Circuit Upgrade Dialog Box**



### Tab and Group Customizations

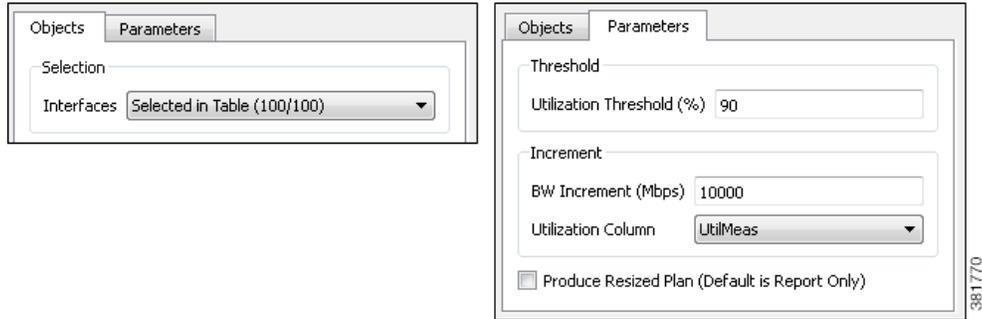
This example `<AddOnOptions>` table customizes the default sample `addon.txt` file as follows (Figure 7-3).

- Creates two tabs: Objects and Parameters
- In the Objects tab, creates a group named Selection.
- In the Parameters tab, creates two groups: Threshold and Increment. Another available selection is created to give the option to produce the resized plan file is left ungrouped. A resized plan is an output plan that identifies which circuits need to be upgraded; they are resized based on the criteria set. If not used, you only receive a report detailing the actions WAE Design suggests making.

#### `<AddOnOptions>`

Tab	Group	OptionName	Prompt	Type	Default	Range	Table
Objects	Selection	interfaces	interfaces	TableSelection			Interfaces
Parameters	Threshold	thresh	Utilization Threshold (%)	Float	90	0;100	
Parameters	Increment	increment	BW Increment (Mbps)	Float	10000		
Parameters	Increment	col	Utilization Column	DropDown	UtilMeas	UtilMeas;Util-Sim;WCUtil	
Parameters		resize	Produce Resized Plan (Default is Report Only)	CheckBox	FALSE		

Figure 7-3 Example Customizable Tabs and Groups

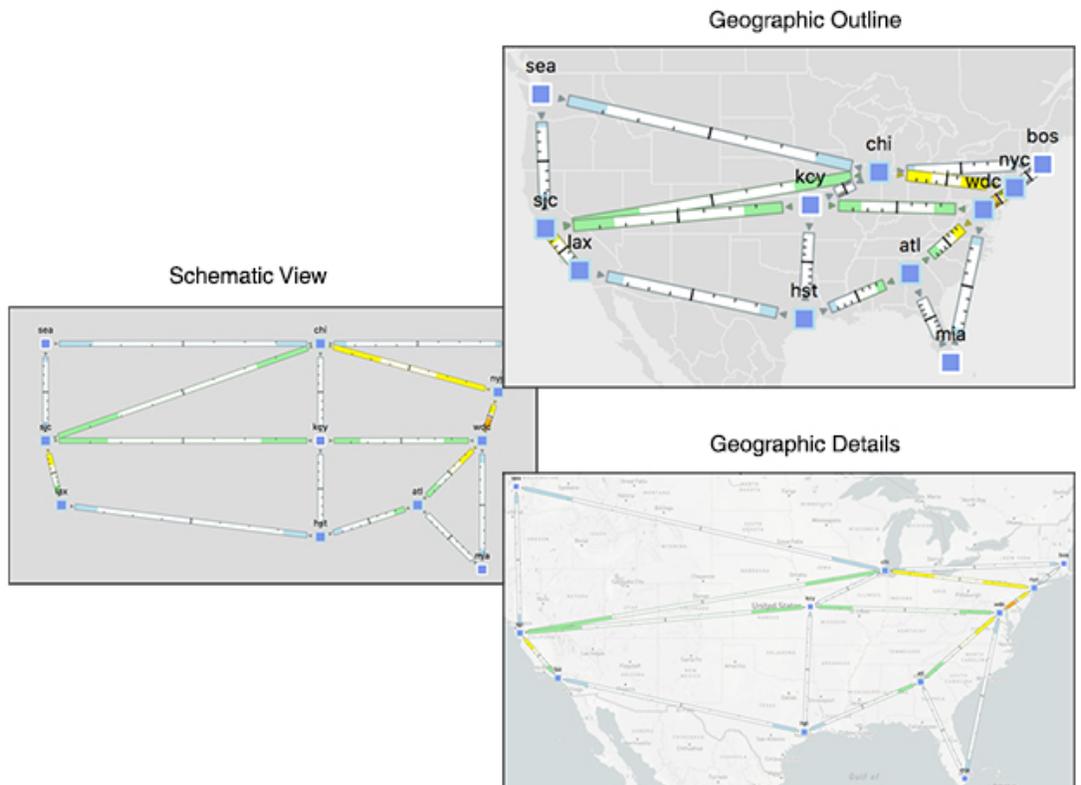


## Customized Network Plot Views

### Map Server

WAE Design uses an online detailed geographic map database to draw the detailed map background, which is one of the canvas plot options available in the WAE Design GUI.

For information on viewing a detailed background map and changing plot options, see the *Cisco WAE Network Visualization Guide*.



## Static Backgrounds

By editing the plan file directly (for example, through an add-on), you can add a user-defined static view to display interfaces with customized color fills and fill percentages. The view then appears as an option in the Network Plot drop-down list located in the far left corner of the visualization toolbar.

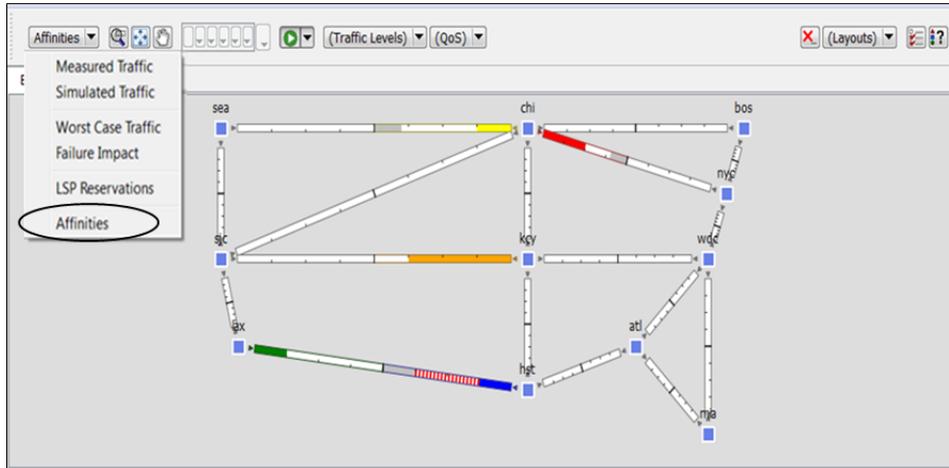
To create this custom network plot view, create or edit a `<PlotViewInterfaces>` table (Table 7-5). You must have an entry (row) for each interface you want you want color filled, and each of these interfaces must have the same PlotView name. For example, Figure 7-4 shows four interfaces in the user-defined plot view named Affinities.

**Table 7-5** `<PlotViewInterfaces>` Columns

Column	Description
PlotView	Name of the plot view; this name appears in the Network Plot drop-down list.
Node	Node containing the interfaces.
Interface	Interface name. Example: POS1/1/1
FillPercent	The percent of the interface to be filled with color.
FillBound	Used only if the FillPercent color represents a utilization percentage. FillBound is the maximum utilization percentage possible, or the QoS bound. The remainder of the interface is gray. The default is 100%. <ul style="list-style-type: none"> <li>• FillBound less FillPercent is colored white. For example, on cr1.chi to cr2.nyc in Figure 7-4, the FillBound is 80 and the FillPercent is 50, thus filling 30 percent of the interface with white.</li> <li>• If the FillBound is greater than the FillPercent, then the difference is filled with red and white stripe, which represents a QoS violation. For example, on the cr2.hst to cr1.lax interface in Figure 7-4, the FillBound is 25 and the FillPercent is 75, thus filling 50% of the interface as a QoS violation.</li> </ul>

**Figure 7-4** Example <PlotViewInterfaces> Table Defining an Affinities View and Coloring Five Interfaces

<PlotViewInterfaces>					
PlotView	Node	Interface	Color	FillPercent	FillBound
Affinities	cr1.lax	{to_cr2.hst}	Green	25	100
Affinities	cr2.hst	{to_cr1.lax}	Glue	75	25
Affinities	cr1.chi	{to_cr2.nyc}	Red	50	80
Affinities	cr1.chi	{to_cr1.sea}	Yellow	25	80
Affinities	cr1.kcy	{to_cr1.sjc}	Orange	75	100



381769



## Reporting Tools

---

The `manage_reports` tool provides a way to manage reports that are generated in a plan file.

### Reports in Plan Files

Reports are created when certain tools are run, such as Simulation Analysis (`sim_analysis`) and Demand Deduction (`dmd_deduct`). These reports are stored in the plan file on which the tool is run and can be accessed by choosing **Window > Reports**. Add-ons and other scripts can use this reporting functionality.

The `manage_reports` tool provides access to the reports in a plan file for any of the following functions:

- Insert a report into a plan file
- Delete a report from a plan file
- Extract a report from a plan file
- Rename a report within a a plan file
- List the reports in a plan file
- Print a report in a plan file

After a report is generated, use the `manage_reports` tool to insert the report into a plan file. If `manage_reports` is used to extract a report from a plan file, the extracted report uses this same report format.

### Plan File Report Format

The plan file report format consists of a directory containing the data and formatting details for each section in the report. The directory must contain a `config.txt` configuration file that includes a `<Sections>` table identifying sections of the report. All columns in this `<Sections>` table are required ([Table 8-1](#)).

Optionally, the `config.txt` file contains a `<TableColumns>` table that provides formatting information on how to display the TABLE sections of the report ([Table 8-2](#)). If the `<TableColumns>` table is not included, WAE Design defaults are used.

The name of the report is the name of the directory containing the report. If you create a report in an add-on executable using the `report-dir` option, in which case you cannot name the directory, the report is named after the add-on.

**Table 8-1** Columns of the Required <Sections> Table in the config.txt File

Column	Description
Name	Section name that appears on the report.
Type	Output format of the report: HTML, TEXT, or TABLE.
Filename	The file in the report directory containing the source data for the section. <ul style="list-style-type: none"> <li>• If the ContentType is HTML, this file must contain HTML.</li> <li>• If the ContentType is TEXT, this file must contain plain text.</li> <li>• If the ContentType is TABLE, this file must contain WAE Design tables. One of those tables must use the same name as the section name (Name column) of this row. This table is used as the contents of this section.</li> </ul> <p>Example: If the Name column of this row is Interfaces, this file must contain a table named &lt;Interfaces&gt;.</p>
Index	An integer representing the order in which the sections appear in the report.

**Table 8-2** Columns in the Optional <TableColumns> Table in the config.txt File

Column	Description
Table	The name of the section being defined. This name must be the same as a TABLE section name in the <Sections> table.
Column	The name of the column.
DisplayName	Optional: Alternative column name. If not specified, the Column name is used. Example: You could set the column name to TraffMinusReservation, and the display name to Traffic Reservation.
Shown	Optional: Whether the column is shown when the plan file opens (T) or hidden (F, default). If a report is imported into the plan using an add-on's report-dir option or through manage_reports, then the Shown setting is copied from that report.
ToolTip	Optional: The information to be displayed when you hover the cursor over the column heading in the GUI.
Type	Optional: The sorting order for the column: REAL, TEXT (default), INTEGER, or BOOLEAN.
Decimals	Optional: If the column Type is REAL, this value specifies the number of decimal places to use. The default is empty, which means do not constrain the decimal places.

## Example Plan File Report Directory and Tables

The following example shows a sample report in the /QuarterlyReport directory:

```
% ls QuarterlyReport/
config.txt
summary.txt
file.txt
log.txt
```

Table 8-3 shows the <Sections> table in the config.txt file. Notice that each section name has an associated file that corresponds with the files listed in the above directory. Both the Interfaces and the Nodes sections of this report draw their contents from the same file.txt file. The report output lists these sections in the order identified in the Index column.

**Table 8-3** Example <Sections> Table

Name	Filename	Type	Index
Summary	summary.txt	HTML	1
Interfaces	file.txt	TABLE	2
Nodes	file.txt	TABLE	3
Log	log.txt	TEXT	4

Table 8-4 shows the columns of the Nodes and Interfaces TABLE sections being defined in the <TableColumns> table in the config.txt file. All empty fields use the WAE Design defaults.

**Table 8-4** Example <TableColumns> Table

Table	Column	DisplayName	Type	Decimals	Shown	ToolTip
Node	Name					
Node	Status					Upgrade, remove, or keep
Interfaces	Capacity		REAL	2		Required capacity
Interfaces	Node					Interface node
Interfaces	Active		BOOLEAN		F	Is it running?
Interfaces	Interface	Interface Name				





# Command-Line Interface

The CLI tools allow scripting of various functions performed by the WAE Design GUI, such as import, export, initialization, simulation, and optimization. CLI tools can, in some cases, provide additional options that are not available from the GUI, such as network interface tools. This chapter describes the CLI options file and logging levels.

- [CLI Options File](#)—Format of the `options.txt` file, which can be used to set default options for use by calls to the tools through the CLI or through the GUI.
- [Logging Levels](#)—Options that control the level of logging.

## CLI Options File

Options for any tool can be invoked by specifying these options at the CLI, or by specifying the options in an `options.txt` file. By default, each CLI tool looks for this options file in the `$CARIDEN_HOME/etc` directory. A different options file can be specified when calling any of the tools by using the `-options-file` option.



**Note**

Options specified at the command line take precedence over options specified in the options file.

The Options file format consists of zero or more lines of the following form:

```
<tool-name>:<option>=<option-value>
```

Here `<tool-name>` is the tool name; `<option>` and `<option-value>` are an option and its value, specific to that tool.

Lines starting with the character `#` are treated as comments and ignored.

If a filename is specified as a parameter to an option in `options.txt` using a relative path or without a path, the path will be relative to the directory in which the tool is called, and not the directory containing the `options.txt` file.

When tools are called through the GUI, the default options displayed in the dialog correspond to the options file settings. Note that changes in the options dialog entries will however remain in effect for the remainder of the session, or until changed.

Options to each command-line tool are available by calling the tool with the `-help` option.

# Logging Levels

Many commands have an option (-verbosity) that controls the level of logging.

**Table 9-1** Log Levels

Level	Type	Description
0	Fatal	Premature termination of a CLI tool or GUI process under circumstances beyond our control, so output might be corrupt. Minimal effort to handle it elegantly. Example: disk full, cannot allocate memory.
1-10	Error	An error occurred so that the results of the tool might be wrong, but there will still be output. Probably premature termination. Example: couldn't log in to the seed router for snmp_find_igp_db, bad import format, bad .pln file format.
11-20	Warning	Use of deprecated API, poor use of API, unexpected or undesirable situations. While having more information to continue is preferable, a best guess is made. Examples: Replacing unsupported characters on import, updating SRLG table format on import, unequal capacities on import.
21-30	Notice	Results of a tool that can be used to make a decision about what to do next. Examples: which routers could not find communities.
31-40	Info	Useful information to further assist users. Can be per object. Timing information for long processes. Examples: Metric Optimization completed in 45 seconds, Tried to get community for node A, succeeded.
41-50	Debug	Detailed information of flow through system to help trace back an error or warning. SNMP queries: router needed to be retried.
51-60	Trace	Detailed information of flow through system to help trace back an error or warning. Examples: individual SNMP queries, Full Mosek output for demand deduction, metric optimization.