Plan Files and Tables

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

- Plan Files
- Plan Tables
- Working With Tables
- External Tables

Plan Files

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

WAE Collector creates plan files when collecting from a network and makes them available for use by applications, such as WAE Design and WAE Live, as well as by other WAE modules. 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. The default format for the GUI is the .pln format file.

.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. For a description of the tables in the .pln format file, refer to the External Tables section.

.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 and <Nodes> and <Sites>. Rows starting 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 the External Tables section. 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 table 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.

Sample Simple .txt Format Plan File

To view sample tables in a simple .txt format plan, open the `samples/euro4.txt` file in Excel, or any text file editor. Following are a few `euro4.txt` excerpts for some of the most important tables in the plan.

Table 2-1 <Network>

Property	Value
Title	
Version	5.3

Note If the Version row is missing, the version property defaults to 4.1, not the current version, for backward compatibility. Plans in Version 4.1 did not require an explicit version.

Table 2-2 <Nodes> Excerpt

Name	Site	Function	Protected
cr1.ams	AMS	core	F
er1.fra	FRA	edge	F

Table 2-3 <Sites> Excerpt

Name	LocationCode	Longitude	Latitude
AMS	AMS	4.78	52.32
FRA	FRA	8.57	50.03

Table 2-4 <Interfaces>Excerpt

Node	Interface	IGPMetric
cr1.ams	{to_cr2.ams}	10
er1.fra	{to_er1.lon}	100

Table 2-5 <Circuits>Excerpt

Name	NodeA	InterfaceA	NodeB	InterfaceB	Capacity
PAR-LON	cr1.lon	{to_cr1.par}	cr1.par	{to_cr1.lon}	622
AMS_INTRA	cr1.ams	{to_cr2.ams}	cr2.ams	{to_cr1.ams}	2488
AMS_EDGE1	cr1.ams	{to_er1.ams}	er1.ams	{to_cr1.ams}	

Table 2-6 <Demands> Excerpt

Name	Source	Destination	ServiceClass
er1.lon-er1.ams	er1.lon	er1.ams	Internet
er1.lon-er1.ams	er1.lon	er1.ams	Voice

Table 2-7 <DemandTraffic> Excerpt

Name	Source	Destination	ServiceClass	TrafficLevel	Traffic
er1.lon-er1.ams	er1.lon	er1.ams	Internet	morning	200
er1.lon-er1.ams	er1.lon	er1.ams	Voice	evening	138

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, select the File->Save As menu.

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. For example, `mate_convert` can upgrade a 5.6 plan file to the 6.0 file format.

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 online listing of tables and their columns, refer to the `table_schema.html` file in the `$CARIDEN_HOME/docs` directory. You can also refer to the *WAE Design Plan Table Schema and CLI Reference*, which lists the most commonly used columns of each of these plan tables.

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 2-8 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 2-8 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

Object	Format
Interface	if{Node | Interface}
IP address	ip{ipaddress} Can reference multiple objects of the same or different type. WAE first attempts to find a node with this loopback IP. If 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{L1 Node A | L1 Node B | Name}
Layer 1 circuit path	l1ctp{L1 Node A | L1 Node B | L1 Circuit | Path Option}
Layer 1 link	l1lnk{L1NodeA | L1NodeB | Name}
Layer 1 node	l1nd{Name}
Node	nd{Name}
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 2-9 lists a table schema excerpt for the <NamedPaths> table. Table 2-10 shows an example of a <NamedPaths> table that has been populated within a plan file.

In Table 2-10, the first column is Name, which is described in the first row of Table 2-9. 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 2-9 NamedPaths Table Schema

Plan File Name	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 2-10 <NamedPaths> Example

Name	Source	Active
PathA	Router1.Vostock	T
PathB	Router1.McMurdo	T

User-Defined Tables and Columns

You can add new tables of your own design and new columns to the standard tables. Using these 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.

WAE Design does not check them for errors. However, they are preserved on import and export, and their contents are displayed in the Plan Window under the Plan Tables hierarchy.

These user-defined structures use namespaces, which is a convenient way of grouping tables or grouping columns under one name. Each user-defined table or column is prefixed with a namespace.

The format of the namespace prefix is as follows. You only need one Namespace, though you can chain them together using :: between their names.

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

<Namespace1::Namespace2:: ... ::ColumnName>

The following example shows a new table that defines rack space for nodes. The table title is prefixed with MyTables::, which identifies it as a user-defined table in the MyTables namespace.

Table 2-11 Example: User-Defined <MyTables::Rackspace> Table

Node	Cabinet	RUs
Node1	123	2
Node2	149	1

The next example shows how to add a user-defined column to a standard <Sites> table. The new column is prefixed with Customer::, which identifies it as a user-defined column in the Customer namespace.

Table 2-12 Example: User-Defined Column

Name	Customer::Type
Site1	Datacenter
Site2	Customer

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

This section explains how to 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, inconsistent, or have erroneous data. Such problems are flagged the next time the plan is opened in WAE Design.

See also the Plan Tables and SQL Queries in WAE Design sections.

Add Table in a Plan

Step 1 Create the desired table as a text format file, or extract one from another plan file. If you create a table manually, the table name and columns must confirm to the table definitions in this chapter.

Step 2 Add the table to the plan using `table_replace`. This overwrites an existing table if one exists.

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 using `table_edit`.
- Apply changes with a complex SQL statement 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, 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.

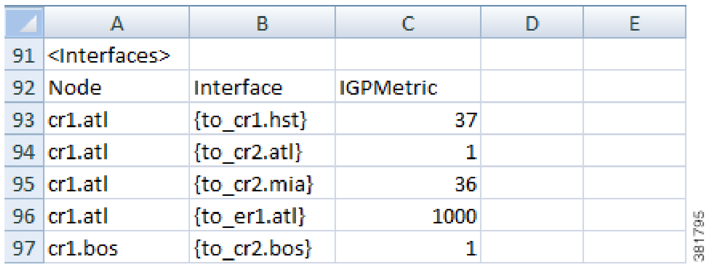
Delete Table from a Plan

Delete the table from a plan using `table_delete`.

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		

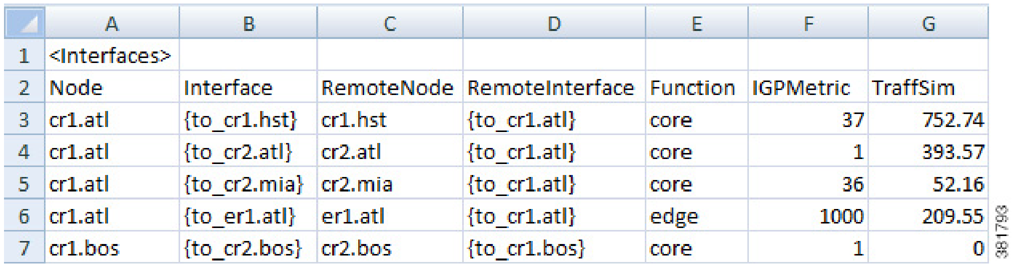
381795

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

Step 1 First, 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

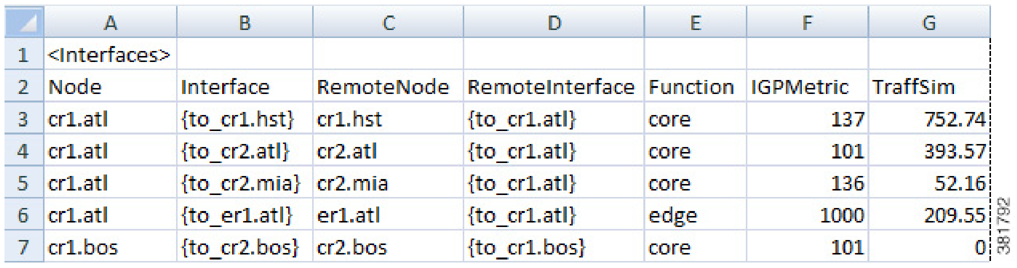
381793

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 is 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

381792

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 just 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 As a confirmation, you could open the `us_wan.pln` file just created to verify the updates. The GUI table would contain the same information shown in Figure 2-3.

Edit Plan File Tables Using the GUI

Step 1 Select the Edit->Plan Tables as Database menu. The Plan Table Database Editor dialog box 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 on the field you want to change and then edit the information, or choose one of these more advanced editing options.

- Select the Set Value tab, choose the Column to modify, enter an SQL Lite expression that specifies the change, and click Set.
- Select the Advanced Filter tab, and 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) IPAddress column 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 First, 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_lsps.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.

Step 1 `mate_select`—Filters tables in the reports.

Step 2 `mate_summary`—Summarizes tables in the reports, primarily to provide summary data for network visualization over time, using the archive feature.

Step 3 `mate_sql`—An advanced SQL query tool.

WAE Design uses the SQLite implementation of the SQL language (see www.sqlite.org, especially 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 you could 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)"
```

WAE Design Perl Library

WAE Design provides Perl modules that simplify access to files that use the plan schema table file format. This format is used, for example, in exported plan files and report files generated from the GUI. The Perl modules are contained in the `$CARIDEN_HOME/lib/perl` directory. Documentation is available using perldoc. For example, execute the following command from the installation directory.

```
perldoc lib/perl/MATE
```

Note To run a Perl script in a Windows installation, a Perl implementation needs to be present. We recommend ActivePerl, which can be downloaded for free.

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.

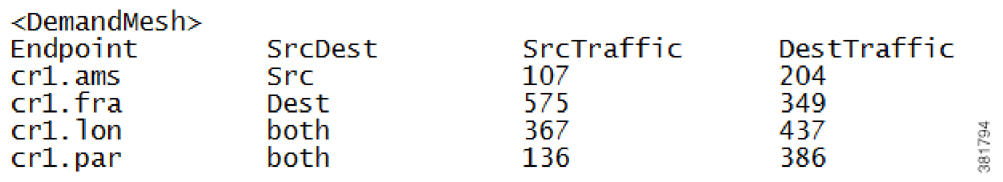
- <DemandMesh> Table
- <Xref><Edits> Table
- Tables for importing traffic and traffic growth rates; see the Importing Traffic and Growth Rates chapter.
- Tables for exporting routes and explicit LSP path settings; see the Exporting Routes chapter.

<DemandMesh> 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 2-3).

- EndPoint—Name of the demand's source or destination, for example, cr1.ams or AS{1234}.
- SrcDest—Identifies whether the endpoint is a source (Src), destination (Dest), or both. destination mesh. Default is Src. Note that an endpoint can appear twice, once in each mesh, in which case the SrcTraffic and DestTraffic should be summed. If a blank entry is summed with a non-blank entry, it becomes blank.
- SrcTraffic—Source traffic. If blank, traffic is estimated based on others and demand mesh settings
- DestTraffic—Destination traffic. If blank, traffic is estimated based on others and demand mesh settings.

Figure 2-4 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
```

<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::Name`.
- (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 2-13 Original <Demands> Table

Source	Destination	GrowthRate
A	B	1
A	C	1
B	A	1

Table 2-14 <Edits> Table

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

Table 2-15 Updated <Demands> Table

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