# Migration Strategy

1.1 Preupgrade Considerations
   - Understanding Optimization Goals
   - Resource Recommendations for SAP ASE 15.0
   - Incorporating Statistics in SAP ASE 15.0
   - Recommended Testing Before Upgrade

1.2 Migrating to SAP ASE 15.0 Features
   - Upgrading, and Using New Features Immediately
   - Upgrading, and Using New Features Later
   - Upgrading, But Not Using New Features

1.3 Troubleshooting
   - Query Processing Tips
   - Information to Capture Before Contacting Product Support

# QPTune

2.1 Set up Your System
2.2 Use QPTune to Fix Missing Statistics
   - Start QPTune to Fix Missing Statistics
   - Collect Statistics
   - Fix Statistics
   - undo_fix_stats

2.3 Use QPTune to Tune Queries or Applications
   - Start QPTune to Tune Queries or Applications
   - Collect Metrics
   - Compare Metrics
   - Apply the Best Results

2.4 Configuration File
2.5 Fixing Missing Statistics Using QPTune
   - Using QPTune Custom Modes

2.6 Upgrade Issues
   - Using QPTune While Migrating

2.7 Localization
2.8 QPTune Reference Information

# Run the Query Processor in Compatibility Mode

3.1 Enable Compatibility Mode
3.2 Feature Support in Compatibility Mode
1 Migration Strategy

Migration Strategy outlines strategies for users who want to upgrade and use new features immediately, and for users who would like to upgrade now and use the new features later.

SQL is a free-form language. There are no rules about the number of words you can put on a line or where you must break a line. However, for readability, all examples and most syntax statements in this manual are formatted so that each clause of a statement begins on a new line. Clauses that have more than one part extend to additional lines, which are indented. Complex commands are formatted using modified Backus Naur Form (BNF) notation.

Table 1: Font and syntax conventions

<table>
<thead>
<tr>
<th>Element</th>
<th>Example</th>
</tr>
</thead>
</table>
| Command names, procedure names, utility names, and other keywords display in sans serif font. | `select`  
`sp_configure` |
| Database names and datatypes are in sans serif font.                   | `master database`                                                      |
| Book names, file names, variables, and path names are in italics.      | `System Administration Guide`  
`sql.ini file`  
`<column_name>`  
`$SAP/ASE directory` |
| Variables—or words that stand for values that you fill in—when they are part of a query or statement, are in italics in Courier font. | `select <column_name> from <table_name>`  
`where <search_conditions>` |
<p>| Type parentheses as part of the command.                               | <code>compute &lt;row_aggregate&gt; (&lt;column_name&gt;)</code>                             |
| Double colon, equals sign indicates that the syntax is written in BNF notation. Do not type this symbol. Indicates “is defined as”. | <code>::=</code>                                                                      |
| Curly braces mean that you must choose at least one of the enclosed options. Do not type the braces. | <code>{cash, check, credit}</code>                                                     |
| Brackets mean that to choose one or more of the enclosed options is optional. Do not type the brackets. | <code>[cash | check | credit]</code>                                                      |
| The comma means you may choose as many of the options shown as you want. Separate your choices with commas as part of the command. | <code>cash, check, credit</code>                                                      |</p>
<table>
<thead>
<tr>
<th>Element</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>The pipe or vertical bar (</td>
<td>) means you may select only one of the options shown.</td>
</tr>
<tr>
<td>An ellipsis (...) means that you can repeat the last unit as many times as you like.</td>
<td>buy thing = price [cash</td>
</tr>
<tr>
<td></td>
<td>You must buy at least one thing and give its price. You may choose a method of payment: one of the items enclosed in square brackets. You may also choose to buy additional things: as many of them as you like. For each thing you buy, give its name, its price, and (optionally) a method of payment.</td>
</tr>
</tbody>
</table>

- Syntax statements (displaying the syntax and all options for a command) appear as follows:

  ```sql
  sp_dropdevice [<device_name>]
  ```

For a command with more options:

  ```sql
  select <column_name>
  from <table_name>
  where <search_conditions>
  ```

In syntax statements, keywords (commands) are in normal font and identifiers are in lowercase. Italic font shows user-supplied words.

- Examples showing the use of Transact-SQL commands are printed like this:

  ```sql
  select * from publishers
  ```

- Examples of output from the computer appear as follows:

<table>
<thead>
<tr>
<th>pub_id</th>
<th>pub_name</th>
<th>city</th>
<th>state</th>
</tr>
</thead>
<tbody>
<tr>
<td>0736</td>
<td>New Age Books</td>
<td>Boston</td>
<td>MA</td>
</tr>
<tr>
<td>0877</td>
<td>Binnet &amp; Hardley</td>
<td>Washington</td>
<td>DC</td>
</tr>
<tr>
<td>1389</td>
<td>Algodata Infosystems</td>
<td>Berkeley</td>
<td>CA</td>
</tr>
<tr>
<td>(3 rows affected)</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

In this manual, most of the examples are in lowercase. However, you can disregard case when typing Transact-SQL keywords. For example, SELECT, Select, and select are the same.

SAP ASE sensitivity to the case of database objects, such as table names, depends on the sort order installed on SAP ASE. You can change case sensitivity for single-byte character sets by reconfiguring the SAP ASE sort order. For more information, see the System Administration Guide.

**If you need help**

Each SAP installation that has purchased a support contract has one or more designated people who are authorized to contact SAP Product Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact SAP Product Support or the SAP subsidiary in your area.
Optimization Goals

QPTune gives a comprehensive summary of the QPTune tool. Run the Query Processor Compatibility Mode, in discusses “the compatibility mode” feature for users who want to upgrade to a new version, but retain performance characteristics of a previous version.

SAP ASE includes a sophisticated query optimizer that analyzes statistics from queried tables, using advanced algorithms to provide better performance. SAP ASE 15.0.3 ESD #1 and later include tools that support more effective use of SAP ASE 15.0’s advanced query optimizer.

This book discusses optimization goals and recommendations for upgrading from SAP ASE 12.5 to SAP ASE 15, analyzing performance differences between the two versions, and tuning SAP ASE 15.0 installations.

This book also documents a tool called QPTune, which enables users to identify and apply the best query plan, optimization goals, and other configuration settings.

1.1 Preupgrade Considerations

This section covers optimization goals and criteria, and the steps you must complete before you upgrade to SAP ASE 15.0.

It includes a summary of preupgrade tests that help you evaluate performance after upgrading your production server.

1.1.1 Understanding Optimization Goals

A central concept of SAP ASE 15.0’s query processing engine is the “optimization goal”, which provides an indication of the nature of the query being optimized.

The SAP ASE query optimizer determines how best to optimize a query based on optimization goals.

For example, a typical OLTP (online transaction processing) query and a typical DSS (decision-support system) query result in very different query plans due to the different data access patterns used by these queries. OLTP queries generally affect only one or a few rows and join only a few well-indexed tables. However, DSS queries typically affect many rows, return a few rows, and may join many tables. Because of their different access patterns, OLTP queries often run most efficiently using a classic “nested-loop join”, whereas DSS queries are more likely to run faster with a “hash join”. If you indicate that a query is for OLTP or DSS purposes, the optimizer uses that information to generate a query plan that may save time, memory, and CPU usage.

SAP ASE 15.0 provides three optimization goals, ordered from “narrow” to “wide,” which correspond to the number of options and strategies that they allow the optimizer to consider:

- allrows_oltp – is best for OLTP queries. allrows_oltp offers the narrowest selection of join methods: the query optimizer considers only nested-loop joins.
- allrows_mix – is the default after upgrading to SAP ASE 15.0. allrows_mix allows the optimizer to consider merge joins as well as parallel plans (if the SAP ASE is configured for parallelism).
• `allrows_dss`—is best for DSS queries. `allrows_dss` offers the widest selection of join methods. The optimizer considers hash joins, as well as nested-loop joins, merge joins, and parallel plans.

If you use `allrows_mix` and `allrows_dss`, additional low-level processing algorithms are enabled for SQL operations; these algorithms are disabled if you use `allrows_oltp`. When you widen the optimization goal, the query optimizer might use significantly more resources (time and procedure cache) to generate a query plan. If the optimizer generates the same query plan, with only nested-loop joins, under `allrows_dss` and `allrows_oltp`, you may expect the optimization under `allrows_dss` to take more time and procedure cache than under `allrows_oltp`.

The choice of optimization goal can have a significant impact on query performance. If you know that a certain application has different workload characteristics than the rest of your system, you may want to set an appropriate session-level optimization goal for that application. Either use the QPTune utility, or manually experiment with different optimization goals, and select one that provides the best overall performance for your particular set of applications and queries.

You can define the optimization goal at the server-, session- or individual-query level:

- **Server-wide default:**
  
  ```
  sp_configure 'optimization goal', 0, 'allrows_dss'
  ```

- **Session-level setting (overrides server-wide setting):**
  
  ```
  set plan optgoal allrows_dss
  ```

- **Query-level setting (overrides server-wide and session-level settings):**
  
  ```
  select * from T1, T2 where T1.a = T2.b
  plan '(use optgoal allrows_dss)'
  ```

**Note**

You can also use a **login trigger** to set the session-level optimization goal.

---

**Related Information**

QPTune [page 18]

---

### 1.1.1.1 Optimization Criteria

An optimization goal is a collection of “on/off” settings for a series of properties known as “optimization criteria.”

Optimization criteria allow or disallow the optimizer to consider a particular algorithm for access methods, joins, grouping, sorting, and so on.

For example, to enable hash joins, use the optimization criterion:

```
set hash_join on
```
Or, to disable the “store_index” algorithm (reformatting), use the optimization criterion:

```
set store_index off
```

The optimizer may decide to ignore a given criteria or goal for semantic reasons. For example, if a user disables all `join` operators, the new optimizer enables “nested loop” automatically.

**Note**

SAP recommends that you use optimization goals, instead of explicit settings for optimization criteria, unless advised to do otherwise by SAP Technical Support.

### 1.1.1.1 Parallel Query Processing in SAP ASE 15.0

Since version 11.5, SAP ASE has supported parallelism within queries, whereby a single query is processed by multiple worker processes.

You can use parallelism to improve response times for DSS-type queries, where a large number of rows are accessed, but only a small result set is returned.

Since the query processing features in SAP ASE 15.0 offer potential performance benefits for DSS-type queries, SAP recommends that, when you upgrade to SAP ASE 15.0, you do not initially use parallel processing.

Serial processing is more resource-efficient than parallel processing although parallel processing allows you to deliver better overall performance with the same hardware. Also, SAP ASE 15.0 in serial mode runs queries faster than earlier versions of SAP ASE with parallelism.

However, parallelism may deliver better response times than serial processing for queries that use semantic table partitioning in SAP ASE 15.0, or for DDL commands such as `create index`.

### 1.1.2 Resource Recommendations for SAP ASE 15.0

SAP ASE 15.0 requires more procedure cache than version 12.5. This increased memory requirement applies to optimization as well as to query execution.

SAP recommends that you increase your procedure cache 2 – 6 times the size of your procedure cache in SAP ASE 12.5.

You may also need to increase the space on `tempdb` for query processing on SAP ASE 15.0.

### 1.1.3 Incorporating Statistics in SAP ASE 15.0

SAP ASE uses a cost-based query optimizer to choose the best plan for a particular query.

The optimizer estimates the cost of different plans based on statistics about the tables, indexes, partitions, and columns referenced in a query. Cost is computed in terms of I/O and CPU time. The optimizer then
chooses the query plan method that has the lowest cost. Inaccurate statistics lead to incorrect cost estimates, and may result in a suboptimal choice of plans and reduced performance.

Some statistics, such as the number of pages or rows in a table (stored in \texttt{sysstabstats}), are updated automatically during query processing. Other statistics are updated only when \texttt{update statistics} runs, or when indexes are created. Examples of this are the histograms on column and density information, stored in \texttt{sysstatistics}.

SAP ASE 15.0 is more susceptible to incorrect statistical data than earlier SAP ASE versions, because multiple algorithms are used for sorting, grouping, unions, joins, and other operations. In addition, SAP ASE 15.0 uses statistics in more ways than in SAP ASE 12.x. For example, SAP ASE 15.0 uses statistics to determine the join order in multitable queries.

SAP recommends that you maintain up-to-date histograms for all columns referenced in \texttt{where} clauses, both when the \texttt{where} clauses are used as join predicates and as search arguments. Use the statistics advisor in QPTune to identify critical and missing statistics.

### 1.1.4 Recommended Testing Before Upgrade

Before upgrading your production systems to SAP ASE 15.0, gather details about the performance characteristics of your applications in the production environment of the current, pre-15 version of SAP ASE.

Gathering such data provides a statistical basis for performance analysis.

To compare SAP ASE 12.x and 15.0 performance, run:

- Tests for as many application functions as possible, especially the most critical ones. For each function, measure the response time or throughput. If possible, perform these measurements for each query executed by the application.
- Performance measurements in your current SAP ASE 12.x production system.
- The same function and performance measurement tests in a fully configured “test” system running SAP ASE 15.0, with a copy of the full SAP ASE 12.x production database, and a realistic workload. Run the same queries as in SAP ASE 12.x, and with the same level of concurrent user activity. Capturing the “performance footprint” of your current SAP ASE 12.x production environment provides a good baseline for any comparisons with SAP ASE 15.0. The measurements you capture should include the number of logical I/O operations, elapsed time, compilation time, CPU utilization, \texttt{showplan} output, and so on. To enable a sensible comparison of performance in SAP ASE 12.x and SAP ASE 15.0, gather performance data at two levels, from:
  - Individual queries in isolation, and with a full workload run by multiple users
  - SAP ASE as a whole, from a server-wide resource usage perspective

Several critical aspects affect performance numbers between SAP ASE 12.x and 15. To avoid misleading performance numbers:

- “Warm up” the cache in the same manner for both SAP ASE 12.x and SAP ASE 15.0 testing.
- Use identical cache/buffer pool configurations.
- Increase the procedure cache in SAP ASE 15.0 to about 2 – 6 times the amount used in SAP ASE 12.x.
- Use similar data device layout and placement, especially for log devices and for \texttt{tempdb}.
- Set up test systems where you can easily restore the original database after each test run, especially when data is modified during testing.
1.2 Migrating to SAP ASE 15.0 Features

You may want to use SAP ASE 15.0 features immediately after upgrade, or you may prefer to use new features later.

The following flowchart depicts the different strategies available for SAP ASE migration:

---

**Figure 1: Flowchart of SAP ASE Migration Strategy**

---

*Note*

You may need to increase the size of the data cache in SAP ASE 15.0.
### 1.2.1 Upgrading, and Using New Features Immediately

To upgrade to SAP ASE 15.0 and use the new features immediately, SAP recommends that you skip setting the compatibility mode, and tune the application using QPTune.

**Procedure**

1. Upgrade your SAP ASE 12.5 and database to SAP ASE 15.0.3 ESD #1 or later, which supports migration using QPTune.
2. Use QPTune to turn on the statistics advisor.
3. Run the application queries. QPTune advises you on what statistics are critical, and creates them. Typically, most queries are tuned at this point.
4. Run QPTune with SAP ASE 15.0 optimization goals.
5. Run the queries with different optimization goals and select the best performing optimization goal.
6. Run the application queries again.
7. Check if there are any queries left to be tuned.
8. Run QPTune for queries that need further tuning.
9. Manually tune the remaining queries. Use traditional techniques of analyzing query plans and either rewriting them to obtain the desired performance, or using techniques such as abstract query plans.

### 1.2.2 Upgrading, and Using New Features Later

If you are going to upgrade to SAP ASE 15.0, and incrementally begin using SAP ASE 15.0 features, upgrade your SAP ASE 12.5 and database to SAP ASE 15.0.3 ESD #1 or later, which supports migration using QPTune.

**Context**

Use `allrows_oltp` as the optimization goal and enable compatibility mode for the upgrade.

When you are ready to use SAP ASE 15.0 features:

**Procedure**

1. Use QPTune to turn on the statistics advisor.
2. Run the application queries. QPTune advises you on what statistics are critical, and creates them. Typically, most queries are tuned at this point.
3. Run QPTune using SAP ASE 15.0 optimization goals.
4. Run the queries with different optimization goals and select the best performing optimization goal.
5. Run the application queries again.
6. Check if there are any queries left to be tuned.
7. Run QPTune for queries that need further tuning.
8. Manually tune the remaining queries using abstract query plans.

Results

Note
You can incrementally migrate stored procedures using the same methodology.

1.2.3 Upgrading, But Not Using New Features

If you are upgrading from SAP ASE 12.5, but are not going to use SAP ASE 15.0 features, use `allrows.oltp` as an optimization goal and enable compatibility mode.

Related Information

Run the Query Processor in Compatibility Mode [page 39]

1.3 Troubleshooting

This section discusses query processing performance, and strategies for addressing optimization problems.

1.3.1 Query Processing Tips

SAP ASE 15.0 offers a greatly improved query processing environment.

However, if query plans or query performance are not what you expect, here are some ways to isolate the problem:

- When using different optimization goals, make sure no cached plans are used: changing the session-level or server-wide optimization goal does not recompile cached plans. For stored procedures, either execute them with recompile, or run `sp_recompile` on one of the tables being accessed. For batches, make sure the statement cache is disabled by running `set statement_cache off` first.
To ensure that a stored procedure is always optimized with a particular optimization goal, regardless of server-wide or session-level settings, use `set plan optgoal allrows_xxx` as the first statement in the stored procedure. This works only on SAP ASE 15.0.2 ESD #2 or later.

If your SQL code from SAP ASE 12.x contains explicitly forced join orders (with `set forceplan`), reexamine the join orders before upgrading to SAP ASE 15.0. Such constructs may prevent you from benefiting fully from the capabilities of SAP ASE 15.0.

With SAP ASE 15.0.1 ESD #2 or later, you can enable two trace flags:

- Trace flag 15307 nullifies the effect of any `set forceplan` statements during query plan compilation.
- Trace flag 15308 nullifies any explicit forcing of indexes, prefetch, parallelism, or buffer replacement strategies

You can set both these trace flags (15307 and 15308) during server start-up, or dynamically enable them using `dbcc traceon`. The effects of both are server-wide and neither affects any query plan properties defined by abstract query plans.

If your system consumes too much space in `tempdb`, use the Monitoring and Diagnostic Access tables to see if any particular session consumes a lot of space in a worktable. Enable the monitoring tables and run the following query:

```sql
select SPID, DBName, ObjectName, PartitionSize
from master..monProcessObject
where DBID = tempdb_id(SPID)
order by SPID
```

Look for sessions that have a large value for `PartitionSize`. Worktables have an `ObjectName` of "temp worktable." Find the corresponding SQL statement for the sessions by issuing queries to `monProcessSQLText` or `monProcessStatement` in the `master` database.

To stop sessions from filling up `tempdb`, and thus affecting other sessions also requiring `tempdb` space, create a resource limit of type "tempdb_space." You may also create multiple temporary databases and assign them to specific users. To check the `tempdb` space used by a single session, use:

```sql
select pssinfo(spid|0,'tempdb_pages')
```

Enable the statement cache and literal autoparameterization settings while running large numbers of identical or similar client-generated SQL queries in Adaptive Server 15.0.1 or later. This does not include stored procedures, or execute-immediate query forms, and the queries may differ only in their search parameters. Overall performance is improved by significantly reducing the time and resources spent on query optimization.

When the statement cache is enabled, a query’s plan is cached so you need not compile an identical query, and thereby save time and resources. The statement cache is enabled server-wide with the configuration parameter `statement cache size`. At the session level, disable the statement cache with `set statement_cache off`.

Literal autoparameterization is enabled server-wide with the configuration parameter `enable literal autoparam`, and at the session level with `set literal_autoparam on`. `enable literal autoparam` applies only when the statement cache is enabled. With literal autoparameterization enabled, caching is extended to almost-identical queries that differ only in a constant value. For example, these two queries are not considered identical:

```sql
select CustName from Customers where CustID = 123
select CustName from Customers where CustID = 456
```
However, they are likely to generate the same query plan. Enabling literal autoparameterization has the effect that the statement cache factors out the constant value in the `where` clause and caches a plan for all queries that look like this:

```sql
select CustName from Customers where CustID= <integer-constant>
```

### Obsolete Optimization Commands in SAP ASE 15.0

Various optimization-related settings from 12.x are no longer relevant in SAP ASE 15.0.

Although the following commands still exist in SAP ASE 15.0, they are relevant only in compatibility mode, and do not have any effect on the SAP ASE 15.0 optimization process:

- `set sort_merge` – this has been replaced by `set merge_join`, optimization goals and the configuration parameter `enable merge join`.
- `set jtc` – join transitive closure is always enabled in SAP ASE 15.0.
- `set table count` – this setting is no longer relevant in SAP ASE 15.0.
- `enable sort-merge join and JTC` – this configuration parameter has been replaced by optimization goals and by the configuration parameter `enable merge join`.
- Start-up trace flags 334 and 384 – these flags enabled merge joins and JTC and are no longer relevant.

SAP recommends that you remove any references to these features from your applications.

### 1.3.2 Information to Capture Before Contacting Product Support

Before contacting Tech Support, gather as many diagnostics statistics as possible, especially when you can reproduce the problem.

#### 1.3.2.1 701 Errors

When a regular query (excluding `update index statistics`) generates a 701 error, it indicates that SAP ASE has exhausted the procedure cache space.

If you are running with the default procedure cache size, increase procedure cache and try again. The general guideline for version 15.0 and later is to use a procedure cache that is 2 – 6 times the size of your 12.5.x procedure cache. In some cases, especially while using the optimization goal `allrows_dss`, your procedure cache may need to be even larger.

If increasing the procedure cache does not resolve the 701 error and you cannot isolate the problem, set up a configurable shared memory dump that includes the procedure cache pages:

```sql
sp_configure 'dump on conditions', 1
```
go

sp_shmdumpconfig 'add', 'error', 701, 1, 'my_dump_directory',null,include_proc

sp_shmdumpconfig adds the error 701 condition to initiate a memory dump. The fourth parameter (1 in the examples above) indicates the number of memory dumps to capture. SAP ASE does not capture additional memory dumps on this condition until SAP ASE is restarted or until you manually reset the counter.

The parameter `<my_dump_directory>` is the name of a directory to hold the memory dump. The file system on which the directory resides should have enough free space to hold the memory dump file, which can be large. Verify the dump conditions currently defined by running `sp_shmdumpconfig` without any parameters. This also shows an estimated size of the memory dump to be captured.

The parameter `<include_proc>` enables procedure cache information to be included in the configurable shared memory dump.

A file name that includes the date and time of the memory dump is automatically generated. Once the memory dump has been captured, reset the system using:

```
sp_shmdumpconfig 'drop', 'error', 701
```

By default, SAP ASE sends the 701 error message to the client. You may also have this message reported in the error log by running:

```
sp_altermessage 701,'with_log',true
```

To stop all configurable shared memory dumps, set `dump on conditions` to 0. Once the memory dump has been captured, open a case with Technical Support and upload the memory dump to the FTP site. Include the output from the SQL statements below which use the monitoring tables within SAP ASE:

```
select * from master..monProcedureCacheMemoryUsage
select * from master..monProcedureCacheModuleUsage
```

The monitoring tables are automatically set up during execution of the `installmaster` script in 15.0.2 or later. The installation process for earlier versions of SAP ASE execute the `installmontables` script. See 15.0 documentation for more details on configuring the monitoring tables.
1.3.2.2 Performance Problems with a Limited Number of Queries

If a limited number of queries are not performing well due to suboptimal query plans or suboptimal resource consumption, install the latest SAP ASE 15.0.x version on your development server.

Procedure

1. Create a script file `sql.txt` containing these commands:

```sql
select @@version
go
select @@optgoal
go
sp_cacheconfig
go
sp_configure 'nondefault' (only if you're running SAP ASE 15.0.2 or later)
go
dbcc traceon(3604)
set showplan on
set statistics time, io, plancost on
set option show long
<your query text>
go
```

i Note

`set option show long` may produce a lot of output for complex queries.

2. Use `isql` to execute `sql.txt` and capture the output in a file:

```bash
isql -Usa -P yourpassword -S YOUR_SERVER_NAME
-i sql.txt -o sql.out
```

Use the `-w` option of `isql` to format the output.

3. Send this information to Technical Support:

- The `sql.txt` and `sql.out` files. If available, include the “fast” (`sql.fast.txt`) and “slow” (`sql.slow.txt`) query plans, and corresponding output files `sql.fast.out`, `sql.slow.out`.
- DDL for the base tables and indexes, which you can generate using the `ddlgen` utility.
- Simulate statistics output for the base tables using `optdiag`:

```bash
optdiag statistics simulate <table-name>
-Usa -P yourpassword -S YOUR_SERVER_NAME
-o <output-file>
```

- A copy of the SAP ASE configuration file. For SAP ASE15.0.2, include the output of `sp_configure 'nondefault'`.
- If the query uses views or stored procedures, then include their SQL source code obtained using `defncopy` or `ddlgen`.
- The output of `sp_monitorconfig 'all'` and `sp_helpsort`.
### 1.3.2.3 System-Wide Performance Issues

If the performance of SAP ASE at the server level is not acceptable, and you are running 15.0.2 ESD #3 or later, you may shut down the SAP ASE and restart it with trace flag 757 set in the RUN_server file.

This is also effective when you experience unusually high levels of CPU usage without a clear cause, while running a multiengine SAP ASE.

If the procedure cache is filled with idle cached plans, and the CPU usage is not high, run the following `dbcc` commands instead. However, using these commands is likely to have a lesser effect than restarting the server.

```
  dbcc traceon(757)
  go
  dbcc proc_cache(free_unused)
  go
```

**Note**

Do not use trace flag 757 in SAP ASE versions earlier than 15.0.2 ESD #3.

### 1.3.2.4 Uploading Diagnostics to Product Support

After you have created diagnostic files, open a case with Technical Support. To upload diagnostics to the SAP FTP site, contact Technical Support for current instructions.
QPTune

QPTune is an SAP ASE utility that is written in Java/XML. It enables users to identify the best query plan, optimization goals, or other configuration settings, and apply them at the server or query level.

This results in optimal performance of subsequent query executions. Once you have identified the best settings for application queries, you can export and apply them to production servers.

QPTune allows users to:

- Fix missing statistics in an application
- Tune an application to find the best optimizer settings for any number of queries
- Selectively apply customized or standard settings to specific queries using user-defined rules

Use QPTune to analyze and compare any number of configuration settings or SAP ASE installations to generate a performance impact analysis report, or to perform plan fixes without degrading SAP ASE’s performance. SAP ASE gathers metrics with simple `select` statements and stores it in stored procedure or statement caches. SAP ASE fixes query plans using DDL statements that have little impact on the overall performance of the system. In addition, QPTune allows different threshold levels for monitoring, thereby reducing the metrics that need to be collected.

2.1 Set up Your System

Before starting QPTune, set the system environment variables:

- `SYBASE_JRE6` or `SAP_JRE6` and `JAVA_HOME` – to the Java runtime installation.
- `SYBASE` – to the latest SAP installation on your machine.
- `SYBASE_ASE` – to the SAP ASE component(directory) of the installation on your machine.

The QPTune executable is named `QPTune` on UNIX and `QPTune.bat` on Windows and is found in:

- `$SYBASE/$SYBASE_ASE/qptune`, on UNIX
- `%SYBASE%\%SYBASE_ASE%\qptune`, on Windows

For complete syntax and reference information on QPTune, see QPTune Reference Information.

To verify your environment and installation, and for information on basic syntax, run QPTune with the `-h` option:

```
QPTune -h
```

Sample output in a Windows environment:

```
QPTune <Version 3.0> Windows/Unix Built: Fri Jan 21 14:00:15 PDT 2009
Syntax:
QPTune [-U <username>] [-P <password>] [-S <hostname:port/database>]
[-A <action
[start|collect(_full)|compare|fix|(start|collect|fix|undo_fix)_stats>]
[-M <mode>] [-T <appTime>] [-i <inputFile>] [-o <outputFile>]
```

Migration Technology Guide
QPTune
Related Information

QPTune Reference Information [page 36]

2.2 Use QPTune to Fix Missing Statistics

Use QPTune to fix or update the missing statistics after you have upgraded a server.

The main steps for using QPTune to fix missing statistics are:

- Start QPTune using the `start_stats` action.
- Run the application, queries, or stored procedure.
- Collect any missing statistics information into a specified XML file.
- Use the `fix_stats` action to update statistics as specified in the above XML file.

Note

Only users with `sa_role` and `sso_role` can run QPTune actions, except for `compare`, which may be run by any user.
The tuning cycle to fix missing statistics is shown here:

![Diagram of tuning cycle]

**Related Information**

- Collect Statistics [page 21]
- Fix Statistics [page 22]
- undo_fix_stats [page 23]

### 2.2.1 Start QPTune to Fix Missing Statistics

Start the utility with the `start_stats` action.

For example:

```
QPTune -A start_stats -S my_host:4816/my_database
```
You are now connected to database: my_database
[INFO] Config: sp_configure 'capture missing statistics', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: delete sysstatistics where formatid =110
You may also use the -c option to specify a configuration file. This extracts server-level configuration settings from the <start_stats> section of your configuration file.

Note
SAP recommends that you never enable capture missing statistics in a production environment.

Related Information
Configuration File [page 29]

2.2.2 Collect Statistics

After preparing the system by running QPTune with the start_stats action, you may begin collecting the missing statistics with the collect_stats action.

You can have QPTune either perform this action immediately, or after waiting for some period of time. This feature enables you to automate the start_stats and collect_stats steps.

collect_stats retrieves missing statistics information from the sysstatistics table for statistics that exceed a specified threshold for count of missing statistics. QPTune consolidates the missing statistics and determines a minimum set of statistics that must be updated.

The -m option indicates the threshold for count of missing statistics. When the statistics for a query have been missed as many times as the threshold value or more, they are collected and exported to an XML file. The default threshold count is 5.

The -o option indicates the output XML file that holds missing statistics. Use the output XML from collect_stats as input to the fix_stats and undo_fix_stats actions.

For example:

QPTune -A collect_stats -m 1 -o missingstats.xml -v -S my_host:4816/my_database

You are now connected to database: my_database
Now collecting missing statistics information from sysstatistics on "Fri Sep 26 10:08:06 PDT 2008".
<?xml version="1.0" encoding="UTF-8"?>
<server url="jdbc:SAP:Tds:my_host:4816/my_database"
file="missingstats.xml"
type="missing stats" datetime="Fri Sep 26 10:08:06 PDT 2008" >
<missingStat id="1">
  <id>1068527809</id>
  <stats>Y(y4,y2)</stats>
  <count>2</count>
</missingStat>
<missingStat id="2">
  <id>1068527809</id>
  <stats>Y(y3)</stats>
  <count>1</count>
</missingStat>
<missingStat id="3">
  <id>1068527809</id>
  <stats>Y(y2,y1)</stats>
  <count>1</count>
</missingStat>
<missingStat id="4">
  <id>1068527809</id>
  <stats>Y(y1)</stats>
  <count>1</count>
</missingStat>
</server>
The missing statistics information is written into XML file: missingstats.xml
[INFO] End config: sp_configure 'enable metrics capture', 0
[INFO] End config: sp_configure 'abstract plan dump', 0
[INFO] End config: sp_configure 'system table', 0
[INFO] End config: sp_configure 'capture missing statistics', 0
Program has restored the data source for metrics collection.
----- QPTune finished executing. -----

2.2.3 Fix Statistics

After collecting missing statistics information into an XML file, you can update the statistics that are equal to, or exceed, the threshold for count of missing statistics specified by the -m option.

Use the fix_stats action to update statistics.

The -i option specifies the input XML file that contains all missing statistics.

You can generate a SQL script for updating statistics without executing the actual updates by using the -N option to indicate "noexec", and the -c option to indicate the output script file. The output file is created with all the generated update statistics statements but the statements are not executed. Generated scripts have a SQL file format. Using the -N option gives you the option of running the SQL script at a later time to optimize your resources.

For example:

```
QPTune -A fix_stats -m 1 -i missingstats.xml
-v -S my_host:4816/my_database
```

You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:14:59 PDT 2008"
-----------------------------------------------------------
Details of statements(s) fixed:
-----------------------------------
Fixed statistics:[Update] Y(y4,y2)
[INFO] Fix Statement = update statistics Y(y4,y2)
Fixed statistics:[Update] Y(y3)
[INFO] Fix Statement = update statistics Y(y3)
Fixed statistics:[Update] Y(y2,y1)
[INFO] Fix Statement = update statistics Y(y2,y1)
Fixed statistics:[Update] Y(y1)
[INFO] Fix Statement = update statistics Y(y1)
----- QPTune finished executing. -----

For example:

QPTune -U sa -P -S my_host:5000/my_database
-A fix_stats -m 5 -i missingstats.xml
-N -o missingstats.sql

2.2.4 undo_fix_stats

To revert fixed missing statistics, use the undo_fix_stats action.

undo_fix_stats deletes the statistics that are specified in an XML file whose missing counts are equal to, or exceed, the number specified by the -m option.

For example:

QPTune -A undo_fix_stats -m 1 -i missingstats.xml
-v -S my_host:4816/my_database

Executing : QPTune -U sa -P [unshown] -S jdbc:SAP:Tds:my_host:4816/my_database -
A undo_fix_stats -m allrows_dss -T 0 -i missingstats.xml -o metrics.xml -f null -
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:20:23 PDT 2008"
-----------------------------------------------------------
Details of statements(s) fixed:
-----------------------------------
Fixed statistics:[Delete] Y(y4,y2)
[INFO] Fix Statement = delete statistics Y(y4,y2)
Fixed statistics:[Delete] Y(y3)
[INFO] Fix Statement = delete statistics Y(y3)
Fixed statistics:[Delete] Y(y2,y1)
[INFO] Fix Statement = delete statistics Y(y2,y1)
Fixed statistics:[Delete] Y(y1)
[INFO] Fix Statement = delete statistics Y(y1)
----- QPTune finished executing. -----
2.3 Use QPTune to Tune Queries or Applications

There is a list of main tasks for using QPTune for application or query tuning.

- Start QPTune using either:
  ○ Simple Start if you are applying standard optimization goal settings.
  ○ Custom Start if you are applying special/custom rules to specified queries.
- Run the application, queries, or stored procedure you are tuning.
- Collect the metrics into specified XML files.
- Compare the sets of metrics you have collected for different optimization goals. This step uses the XML file from the above step as input, and generates a performance comparison report.
- Apply the best results from the comparison to each of the specified queries on the target server.

QPTune’s tuning cycle for applications or queries is shown here:

![QPTune Cycle for Application or Query Tuning](image)

**Related Information**

- Simple Start [page 25]
- Custom Start [page 25]
- Collect Metrics [page 26]
- Compare Metrics [page 27]
2.3.1 Start QPTune to Tune Queries or Applications

The `start` action of QPTune prepares the server with correct server-level configuration settings. If a configuration file is used, the settings are extracted from the `<start>` section of the configuration file. The `<end>` section of the configuration file specifies the settings that enable the system to revert to its original state at the end of QPTune `collect` action.

Related Information

Configuration File [page 29]

2.3.1.1 Simple Start

If you are applying standard optimization goal settings, start QPTune using the simple start method.

```
QPTune -S <host>:/port/<database> -A start
   [-M {allrows_oltp, allrows_dss, allrows_mix}]
```

Use the `-M` option to invoke one of the pre-programmed modes that correspond to the three optimization goals in SAP ASE:

- `allrows_mix`
- `allrows_oltp`
- `allrows_dss` (default)

2.3.1.2 Custom Start

If you are applying custom rules to specified queries, use the custom start method.

```
QPTune -S <host>:/port/<database> -A start -M custom_1
   -i input.xml -l 3 [-v]
```

Use the `-M` option to indicate a custom mode. A custom mode is a group of special rules that are specified in the configuration file under the `<mode>` section. Rules are SAP ASE 15.0 optimization criteria that are applicable at the query level using abstract query plans.

The example above uses a custom mode called `<custom_1>` which may be a combination of rules such as:

- `use optgoal allrows_mix`
• use merge_join off
• use opttimeoutlimit 15

Use the -i option to indicate an input XML file which has been generated by QPTune while applying a standard goal setting during the collect phase. A number of collected metrics files may be compared to generate a file with the best goal settings. The input file contains SQL text for the queries.

Use the -l option along with the -i option to indicate the number of queries that should be applied with these special rules. The queries are counted from the start of the file. The default value of the -l option is 0, which implies that all queries in the input file are applied.

2.3.2 Collect Metrics

After starting the system, run your applications and collect metrics into an XML file.

Use the -o option to specify the output metrics file. The -v option provides a verbose output. The -M option indicates custom or standard modes.

You can collect metrics either:
• Immediately, using the -T 0 option, or
• After <t> minutes, using the -T <t> option.

For example, the command below writes XML into a file named a2.xml. The custom mode is depicted within the <bestmode> tags.

```
QPTune -S <host>:<port>/<database> -A collect -T 0
-o a2.xml -v
```

Program has configured the data source for metrics collection.
Now collecting information from sysquerymetrics on "Tue Feb 19 22:16:04 PST 2008".
<?xml version="1.0" encoding="UTF-8"?>
<server url="jdbc:SAP:Tds:SHANGH1:5000" type="ASE" mode="custom_1"
datetime="Tue Feb 19 22:16:04 PST 2008">
<query id="1">
<qtext> select count(T.title_id) from authors A, titleauthor T 
where A.au_id = T.au_id </qtext>
<elap_avg>300</elap_avg>
<bestmode>custom_1</bestmode>
</query>
</server>

Note
You can use the output XML file from the collect operation as input to compare, fix, or start operations.
2.3.3 Compare Metrics

Once metrics are collected, you can compare different XML files to get the best query optimization goal or criteria for each of the queries.

For example:

```
QPTune -A compare -f a1.xml,a2.xml[,a3.xml..] -d 51,10
-o best.xml -S my_host:5000/my_database
```

The `-f` option specifies a list of two or more collected metrics sample files separated by commas. Use quotes to encapsulate the file name if it contains any spaces.

The `-d` option indicates a threshold percentage and absolute value. A performance improvement beyond the threshold percentage and absolute value is considered “outstanding” during the fix operation. The optimization goal/criteria for those outstanding queries is applied to the server as a plan fix.

The default for the threshold percentage and absolute value pair is “5,5”. If only percentage is specified, the absolute value defaults to 0. Percentage values are between 0 and 100; an absolute value can be any number greater than 0.

The `-o` option specifies the result of the comparison in a file. The file holds the best setting for all the queries being analyzed.

The `-s` option enables sorting the files from largest to smallest. The file with the largest set of queries is used as the basis for comparison.

The following example shows the result of a `compare` operation:

```
Compare all the files: | a1.xml, a2.xml|
Report generated on "Tue Aug 19 21:13:04 PST 2008"
--------------------------------------------------------------------------
File #1: [name= a1.xml  :  mode=allrows_mix]
File #2: [name= a2.xml  :  mode=custom_1]
Query count in File #1 : [mode=allrows_mix]         6
Query count in File #2 : [mode=custom_1]            7
--------------------------------------------------------------------------
Query count improved in File #2: [mode=allrows_mix] 3
Total performance improved [from 422 to 129]: 69 %
Following queries run better in File #2: [mode=allrows_mix]
--------------------------------------------------------------------------
Group 1: improved by no more than 25% [0 queries]
Group 2: improved by 25% to 50% [1 queries]
Query: select count(T.title_id) from authors A, titleauthors T where A.au_id = T.au_id
Average elapsed time (ms): File #1=100  File #2=50   Improvement=50.0%
Outstanding=No
Group 3: improved by 50% to 75% [0 queries]
Group 4: improved by 75% to 100% [2 queries]
Query: select count(*) from titles T, titleauthors TA where T.title_id = TA.title_id
Average elapsed time (ms): File #1=34  File #2=7       Improvement=79.0%
Outstanding=Yes
Query: select au_lname, au_fname from authors where state in ("CA", "AZ")
Average elapsed time (ms): File #1=9   File #2=0       Improvement=100.0%
Outstanding=No
```

The above example shows a comparison between two XML metrics files: `a1.xml` has six queries, and `a2.xml` has seven queries. Comparisons can only be made between the queries that are common to both files. There are three queries that ran faster in `a2.xml`. The improvements are categorized into four groups:
There is one query between 25% and 50% and two queries between 75% and 100%. The queries in Group 2 are marked as "Outstanding=No" which means that based on the threshold of 51%, this query will not be fixed.

While comparing more than two files, QPTune updates the first file with the best from both files, then compares the new file with the third file, and so on.

2.3.4 Apply the Best Results

After getting the results for all queries being analyzed, use the `fix` action to apply the best settings to the queries in the database system.

For example:

```
QPTune -S <host>:<port>/<database> -A fix -i best.xml
```

The `-i` option specifies the queries and their best plans resulting from the comparison.

The `-g` option, when used with the `fix` action, applies the default goal. The default goal is the best `optgoal` setting that most queries used as the best plan using QPTune’s `fix` action. This option only generates plans for queries that do not currently use the server’s default optimization goal.

The example `fix` action above produces this output:

```
Query Plan(s) fixed on "Wed Sep 17 17:44:09 PDT 2008"
--------------------------------------------------------------
Fixed 2 queries using mode "custom_1" with following optimizer settings": '(use optgoal allrows_mix) (use merge_join off) (use opttimeoutlimit 15)'
Fixed 4 query using mode "allrows_mix"
Apply "sp_configure optimization_goal, 0, allrows_mix" as the default optgoal

Details of statement(s) fixed:
----------------------------------------
Query: 'select count(T.title_id) from authors A, titleauthor T where A.au_id = T.au_id '
Fixed using: 'custom_1'
[INFO] Fix Statement = create plan 'select count(T.title_id) from authors A, titleauthor T where A.au_id = T.au_id' '(use optgoal allrows_mix) (use merge_join off) (use opttimeoutlimit 15)'
Query: 'select * from titleauthors where au_id > 20 and title_id < 100'
Fixed using: 'custom_1'
[INFO] Fix Statement = create plan 'select * from titleauthors where au_id > 20 and title_id < 100' '(use optgoal allrows_mix) (use merge_join off) (use opttimeoutlimit 15)'
```

QPTune then creates an optimized query plan which is saved in the `sysqueryplans` system table in the current database. When a query with matching SQL is encountered, this optimized plan is used. Incoming SQL and the SQL of the persistent plan are said to match when a checksum type of hash on the two SQL
statements matches. If literal parameterization is enabled explicitly, the two statements may differ only in the
static values of search arguments such as:

```
where CustomerID = "12345"
```

In this case, the value “12345” is replaced by a placeholder variable, so the hash value is the same, regardless
of the search value.

If the application changes the SQL in any manner, such as adding a new predicate, there is no longer a match
to a persistent plan and the optimizer creates a query plan according to the current configuration and available
statistics.

## 2.4 Configuration File

You can define custom modes in a configuration file. The QPTune installation includes a standard
configuration file that contain some custom modes.

The custom mode “_basic_” is reserved for “basic optimization”.

The configuration file for QPTune must include `<start>`, `<start_stats>`, `<fix>` and `<end>` sections. The `<mode>`
section is optional.

The `<start>` section indicates the configuration settings for SAP ASE before metrics are collected. For
example:

```
<start>
  <!-- Recommended server settings -->
  <start_config>sp_configure 'enable metrics capture', 1</start_config>
  <start_config>sp_configure 'abstract plan dump', 1</start_config>
  <!-- Clean up sysqueryplans & sysquerymetrics tables -->
  <start_config>sp_configure 'system table', 1</start_config>
  <start_config>sp_metrics 'flush'</start_config>
  <start_config>delete sysqueryplans where gid=1 or gid=2</start_config>
  <!-- Optional settings that users can change or remove -->
  <!-- <start_config>sp_configure 'enable literal autoparam', 1</start_config> --
  <!-- <start_config>sp_configure 'metrics elap max', 0</start_config> -->
  <!-- Hint: sp_add_resource_limit can be added to limit resource usage -->
  <!-- Specify a query plan group name to save all existing plans from ap_stdin -->
  <!-- Existing plans from ap_stdout will be saved to the corresponding group name
  added with '_out'. -->
  <save_plans_pre_start>pre_start_qpgroup</save_plans_pre_start>
</start>
```

The `<end>` section corresponds to the `<start>` section and includes the configurations setting to be applied
after metrics are collected. For example:

```
<end>
  <end_config>sp_configure 'enable metrics capture', 0</end_config>
  <end_config>sp_configure 'abstract plan dump', 0</end_config>
  <end_config>sp_configure 'system table', 0</end_config>
  <end_config>sp_configure 'capture missing statistics', 0</end_config>
  <!-- <end_config>sp_configure 'enable literal autoparam', 0</end_config> -->
  <!-- <end_config>sp_configure 'metrics elap max', 0</end_config> -->
</end>
```
The `<start_stats>` section includes statistics settings. For example:

```xml
<start_stats>
<!-- Recommended server settings -->
<start_stats_config>sp_configure 'capture missing statistics', 1</start_stats_config>
<!-- Reset counter of missing statistics -->
<start_stats_config>
sp_configure 'system table', 1
</start_stats_config>
<start_stats_config>
delete sysstatistics where formatid=110
</start_stats_config>
</start_stats>
```

The `<fix_stats>` section includes:

```xml
<!-- The following set of configurations apply at "-A fix" -->
<fix>
<!-- Recommended server settings -->
<fix_config>sp_configure 'abstract plan load', 1</fix_config>
<!-- Clean up sysqueryplans & sysquerymetrics tables -->
<fix_config>sp_configure 'system table', 1</fix_config>
<fix_config>sp_metrics 'flush'</fix_config>
<fix_config>delete sysqueryplans where gid=1 orgid=2</fix_config>
<!-- Optional settings that users can change or remove -->
<fix_config>sp_metrics 'enable metrics capture', 1</fix_config>
<fix_config>sp_configure 'enable literal autoparam', 1</fix_config>
<fix_config>sp_configure 'metrics elap max', 0</fix_config>
<!-- Specify a query plan group name to save all existing plans from ap_stdin -->
<save_plans_pre_fix>pre_fix_qpgroup</save_plans_pre_fix>
</fix>
```

The optional `<mode>` section allows users to specify custom optimization settings to one or more queries specified through another input file. The `-M` option of the start and collect actions specifies the mode setting. When the `-M` option specifies anything other than a standard optimization goal setting, QPTune treats the mode as customized, and retrieves the optimization goal and rules settings, for the indicated name, from the `<mode>` section of the configuration file. QPTune then applies the custom settings to the list of specified queries.

### 2.5 Fixing Missing Statistics Using QPTune

Run QPTune with `start_stats` to prepare the server to collect missing statistics.

**Procedure**

1. Enter:

   ```bash
   QPTune -A start_stats -v
   -S my_host:4816/my_database
   ```
2. Run the client application, stored procedure, or query.

3. Run QPTune with `collect_stats` action to collect statistics that exceed the threshold for count of missing statistics. You may let the utility wait for some period of time (specified by the `-T` option) before collecting the missing statistics information.

```
QPTune -A collect_stats -m 1 -o missingstats.xml -v -S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:SAP:Tds:my_host:4816/my_database
-A collect_stats -M allrows_dss -T 0 -i null -o missingstats.xml -f null -c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n null -v
You are now connected to database: my_database
[INFO] Config: sp_configure 'capture missing statistics', 1
[INFO] Config: sp_configure 'system table', 1
[INFO] Config: delete sysstatistics where formatid =110
Now collecting missing statistics information from sysstatistics on "Fri Sep 26 10:08:06 PDT 2008".
```

```xml
<server url="jdbc:SAP:Tds:my_host:4816/my_database" file="missingstats.xml" type="missing stats" datetime="Fri Sep 26 10:08:06 PDT 2008" >
  <missingStat id="1">
    <id>1068527809</id>
    <stats>Y(y4,y2)</stats>
    <count>2</count>
  </missingStat>
  <missingStat id="2">
    <id>1068527809</id>
    <stats>Y(y3)</stats>
    <count>1</count>
  </missingStat>
  <missingStat id="3">
    <id>1068527809</id>
    <stats>Y(y2,y1)</stats>
    <count>1</count>
  </missingStat>
  <missingStat id="4">
    <id>1068527809</id>
    <stats>Y(y1)</stats>
    <count>1</count>
  </missingStat>
</server>
```

The missing statistics information is written into XML file: missingstats.xml

[INFO] End config: sp_configure 'enable metrics capture', 0
[INFO] End config: sp_configure 'abstract plan dump', 0
[INFO] End config: sp_configure 'system table', 0
Program has restored the data source for metrics collection
----- QPTune finished executing. -----
4. Update statistics that have exceeded or equalled the threshold for count of missing statistics, specified by the \(-m\) option. To fix missing statistics that are specified in the input file `missingstats.xml`, use:

```
QPTune -U sa -P -A fix_stats -m 1 -i missingstats.xml -v -S my_host:4816/my_database
```

Sample output:

```
-M allrows_dss -T 0 -i missingstats1.xml -o metrics.xml -f null
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1 -n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:14:59 PDT 2008"
-----------------------------------------------------------
Details of statements(s) fixed:
-----------------------------------------------------------
Fixed statistics:[Update] Y(y4,y2)
[INFO] Fix Statement = update statistics Y(y4,y2)
Fixed statistics:[Update] Y(y3)
[INFO] Fix Statement = update statistics Y(y3)
Fixed statistics:[Update] Y(y2,y1)
[INFO] Fix Statement = update statistics Y(y2,y1)
Fixed statistics:[Update] Y(y1)
[INFO] Fix Statement = update statistics Y(y1)
----- QPTune finished executing. -----
```

**Note**

If the `fix_stats` action is used with the `-N` option, QPTune does not execute the statements to fix missing statistics, but instead sends them to an output file specified by `-o <output_file>`.

5. (Optional) The `undo_fix_stats` command deletes the statistics specified in the `-i` XML file. The statistics deleted are those that have missing counts exceeding or equal to a number specified by `-m`. To undo the fix of missing statistics in the input file `missingstats.xml`, use:

```
QPTune -U sa -P -A undo_fix_stats -m 1 -i missingstats.xml -v
-S my_host:4816/my_database
```

Sample output:

```
Executing : QPTune -U sa -P [unshown]
-S jdbc:SAP:Tds:my_host:4816/my_database
-A undo_fix_stats -M allrows_dss -T 0
-i missingstats.xml -o metrics.xml -f null
-c config.xml -l 5 -e elap_avg -d 5,5 -m 1
-n null -v
You are now connected to database: my_database
Fix statistics on "Fri Sep 26 10:20:23 PDT 2008"
-----------------------------------------------------------
Details of statements(s) fixed:
-----------------------------------------------------------
Fixed statistics:[Delete] Y(y4,y2)
[INFO] Fix Statement = delete statistics Y(y4,y2)
Fixed statistics:[Delete] Y(y3)
[INFO] Fix Statement = delete statistics Y(y3)
QPTune Utility
Fixed statistics:[1 Delete] Y(y2,y1)
```
2.5.1 Using QPTune Custom Modes

You can run select queries using your own custom modes defined in a configuration file.

Procedure

1. QPTune includes some custom modes like "_basic_" which represents basic optimization of SAP ASE 12.5. For example, the default configuration file `config.xml` contains custom mode "custom1" which allows an optimization goal of `allrows_oltp`, together with the rule `merge_join_off`:

   ```xml
   <!-- "default" custom mode -->
   <mode name="default">
     <optgoal>use optgoal allrows_mix</optgoal>
     <rule>use merge_join off</rule>
   </mode>
   <!-- "_basic_" mode is a reserved system mode. -->
   <mode name="_basic_">
   </mode>
   <mode name="custom1">
     <optgoal>use optgoal allrows_oltp</optgoal>
     <rule>use merge_join off</rule>
   </mode>
   
   <!-- "_basic_" custom mode -->
   <mode name="_basic_">
     <optgoal>use optgoal allrows_oltp</optgoal>
     <rule>use merge_join off</rule>
   </mode>
   
   Sample output:
   
   Report generated on "Fri Aug 29 13:29:17 EDT 2008"
   
   Sorted List By File Size (Desc.)=sp_telco_basic.xml,best.xml
   File #1 : [name=sp_telco_basic.xml : mode=_basic_]
   File #2 : [name=best.xml : mode=best]
   Query count in File #1: 14
   Query count in File #2: 14
   
   Query count improved in File #2: 7
   Total performance improved [from 2441 to 1529]: 37%
   Following queries run better in File #2:
   Group 1: improved by no more than 25% [4 queries]
2.6 Upgrade Issues

QPTune helps you get the best performance when upgrading to SAP ASE 15.0.

If there are queries that do not perform as well as pre 15.0 versions of the server, QPTune allows SAP ASE Enterprise to generate version 12.5.4-like query plans. If these plans run faster than the corresponding version 15.0 query plans, QPTune retains and uses these plans for all subsequent query execution.

2.6.1 Using QPTune While Migrating

Using QPTune while migrating to SAP ASE 15.0.

Procedure

1. Depending on the application, get metrics information for any or all of the three SAP ASE 15.0’s pre-programmed modes (“mix,” “dss,” “oltp”):

   ```sh
   QPTune -A start -M allrows_oltp
   -S my_host:5000/my_database
   
   <Run your query, application, or stored procedure>
   
   QPTune -A collect -M allrows_oltp -T 0 -o oltp.xml
   -S my_host:5000/my_database
   
   QPTune -A start -M allrows_mix
   -S my_host:5000/my_database
   
   <Run your query, application, or stored procedure>
   
   QPTune -A collect -M allrows_mix -T 0 -o mix.xml
   -S my_host:5000/my_database
   
   QPTune -A start -M allrows_dss
   -S my_host:5000/my_database
   ```
<Run your query, application, or stored procedure>

```
QPTune -A collect -M allrows_dss -T 0 -o dss.xml
-S my_host:5000/my_database
```

2. Get metrics information with optimization similar to version 12.5.4 in SAP ASE 15.0:

```
QPTune -A start -M _basic_ -i oltp.xml -l 10
-S my_host:5000/my_database
```

<Run your query, application, or stored procedure>

```
QPTune -A collect -M _basic_ -T 0 -o basic.xml
-S my_host:5000/my_database
```

3. Compare the metrics information:

```
QPTune -A compare -d 10 -o best.xml
-f basic.xml,oltp.xml,mix.xml,dss.xml
-S my_host:5000/my_database
```

4. Fix query plans with the best out of the comparison:

```
QPTune -A fix -i best.xml
-S my_host:5000/my_database
```

5. (Optional) Verify the performance improvement after the plan fixup, re-run the application, and collect the metrics information:

```
QPTune -A collect -T 0 -o new_best.xml
-S my_host:5000/my_database
```

Performing a compare of new_best.xml with any of the other XML output files should establish that new_best.xml gives the best results.

### 2.7 Localization

The QPTune command line utility has been localized so its messages can display in these 9 languages other than English: Chinese, French, German, Japanese, Korean, Polish, Portuguese, Spanish, and Thai.

The language properties files are packaged in the qptune.jar file. QPTune sets the display according to the language set on the system’s default locale.
2.8 QPTune Reference Information

QPTune is an SAP ASE utility written in Java/XML. It enables users to identify the best query plan, optimization goals, or other configuration settings, and apply them at the query or server level.

Description

This results in optimal performance of later query executions.

Syntax

```
```

Example:

```
QPTune -U sa -P -S my_host:5000/my_database -A collect -M allrows_mix -T 0 -o metrics.xml -c config.xml -e elap_avg -d 5,5 -l 5 -i metrics.xml -fa1.xml,a2.xml,a3.xml -v -s
```

Parameters

- `-U <username>`
  specifies the database user name.

- `-P <password>`
  specifies the database password.

- `-S <server>`
  specifies the database server. The database server is denoted by `<host>:<port>/<database>`.

  **Note**

  You must specify the `-S` option while using any QPTune actions.

- `-A <action>`
specifies the action to be taken. One of: start | collect | collect_full | compare | fix | start_stats | collect_stats | fix_stats | undo_fix_stats.

-J <charset>

specifies the character set used to connect to SAP ASE. If this option is not specified, the SAP ASE uses the server's default character set.

i Note

If the installed JRE does not support the server's default charset encoding, you see an error message during the login process. Use the -J option to specify a more generic character set, such as -J utf8.

-M < mode>

specifies the optimization goal or custom mode for an application. One of: allrows_oltp, allrows_dss, allrows_mix. You may also define custom modes; _basic_ is a system reserved custom mode.

-T <appTime>

specifies the application running time, in minutes.

-o outputFile

specifies the output file.

-i inputFile

specifies the input file for the fix, fix_stats, and undo_fix_stats actions. You can also use -i to apply special rules to the specified queries for start for custom modes.

-f fileList

compares a list of files to get the best plans; use commas to separate filenames.

-c configFile

specifies the configuration file.

-l <limit>

specifies a limit on the number of queries that should be analyzed and applied with special rules.

-e <evalField>

evaluation field used for performance comparison.

-d <difference>

specifies the percentage and absolute value difference for performance improvement to be considered outstanding.

-N

used along with fix_stats and undo_fix_stats. -N generates a SQL script with update statistics or delete statistics statements. The update or delete statements are not executed through QPTune. The statements are written into a SQL script that is specified by the -o option.

-n <login>
specifies the user’s login whose query executions are collected and analyzed.

\texttt{-m \textless missingCount\textgreater}

specifies the threshold value for missing statistics. The default value is 5.

\texttt{-v}

specifies verbose mode.

\texttt{-g}

when used along with the \texttt{fix} action, applies the default goal. The default goal is the best optgoal setting that most queries used as the best plan using QPTune’s \texttt{fix} action. This option only generates plans for queries that do not currently use the server’s default optimization goal.

If specific values are not indicated for the parameters, the following defaults are used:

- \texttt{-A:collect}
- \texttt{-M:allrows_dss}
- \texttt{-T:0}
- \texttt{-o:metrics.xml}
- \texttt{-c:config.xml}
- \texttt{-e:elap_avg}
- \texttt{-d:5.5. If only percentage is specified, absolute value defaults to 0.}
- \texttt{-l \textlt{limit\textgreater}}
- \texttt{-m 5}

**Permissions**

Only users with \texttt{sa_role} and \texttt{sso_role} can run actions other than \texttt{compare} on QPTune.
SAP ASE version 15.0 includes substantive changes to the query processor. For most customers, the new query processor provides a faster and more efficient environment.

However, you may have tuned your server and applications based on the more restricted query processor from SAP ASE version 12.5.4 and earlier and find the benefits of the version 15.0 query processor unsuitable in some situations. In that case, use the compatibility mode to upgrade to SAP ASE 15.0 from version 12.5.x but retain performance characteristics similar to version 12.5.x. When you enable compatibility mode, SAP ASE 15.0 uses a query engine similar to the one used in version 12.5.4, and provides an alternative optimization and execution strategy.

### 3.1 Enable Compatibility Mode

On SAP ASE 15.0.3 ESD #1 and later, you can enable compatibility mode at the session or server-wide level.

- **Session level** – use `set compatibility_mode on | off` to enable or disable compatibility mode for the current session.
- **Server-wide** – use `sp_configure 'enable compatibility mode', 1 | 0` to enable or disable compatibility mode for the server.

Setting compatibility mode at the session level takes precedence over the server level.

*enable compatibility mode* is a dynamic configuration parameter; you need not restart SAP ASE for it to take effect. However, you must remove any compiled plans for stored procedures, or ad hoc queries, from the statement cache.

**Note**

`sp_configure` generates warnings to indicate that enabling compatibility mode does not affect cached query plans that are already in the procedure or statement cache.

`sp_configure` also generates warnings if it detects configuration options that conflict with compatibility mode, such as:

- One of `abstract plan dump`, `abstract plan load` or `abstract plan replace` is set.
- `statement cache` and `literal autoparam` are set.
- The histogram tuning factor is set to a value other than 1.
3.2 Feature Support in Compatibility Mode

Once you enable compatibility mode, SAP ASE uses the query processor for all insert, delete, update, and select queries.

The query processor uses either full or partial compatibility mode:

- Full compatibility mode – SAP ASE 15.0 uses an optimization and execution strategy similar to the one used in version 12.5.x.
- Restricted compatibility mode – SAP ASE uses only an optimization strategy similar to the one used in version 12.5.x.

Generally, SAP ASE uses full compatibility mode wherever possible. If it cannot use full compatibility mode, it switches to restricted compatibility mode.

Table 2: Version 12.5 feature support in compatibility mode

<table>
<thead>
<tr>
<th>12.5.4 features with limited support in compatibility mode</th>
<th>Supported in full compatibility mode?</th>
<th>Supported in restricted compatibility mode?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Queries with text and image columns</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Referential integrity</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Inserts that require referential integrity</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Proxy tables</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Round-robin partitions</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Encryption and cipher text</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Queries that include the rand2 function</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Abstract plans, either explicit (with plan clause) or implicit (plan dump or plan load):</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Extended datatypes, such as Java ADT and Java UDF</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>XML functions</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Browse mode</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Parallel hints</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>Parallel sort</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Compatibility mode does not support these SAP ASE 15.0 features:

- Partitioned tables
- `group by` with more than 31 columns
• Scrollable and insensitive cursors
• Commands on computed columns
• Queries that fire “instead-of-triggers”
• Queries executed inside “instead-of-triggers”
• Queries that issue parameterized literals in the statement cache, unless the query includes an
  insert...values command
• Query processing diagnostics used by showplan_in_xml
• Queries that include hash or hashbyte functions
• User-defined functions (SQL UDFs)
• Explicit timestamp inserts (available for SAP ASE version 15.0.2 and later and in Replication Server®)
• SQL-based replication (available for SAP ASE version 15.0.3 and later)
• group by result rows that are wider than the maximum size that fits on a data page
• xmltable function

**Note**
Query plans in compatibility mode are not executed as parallel plans.

### 3.3 Additional Trace Flag for Diagnostics

Trace flag 477 alters compatibility mode.

For every query evaluated, SAP ASE prints this message to the error log, which indicates if SAP ASE used full
compatibility mode to process the query:

```
Compatibility = true | false
```

The message includes the reason if compatibility mode is not chosen, and the query text, if available.

### 3.4 New Stored Procedure sp_compatmode

Use `sp_compatmode` on SAP ASE 15.0.3 ESD #1 and later, to verify if full compatibility mode can be used
effectively.

`sp_compatmode` generates warnings if it detects configuration options that conflict with compatibility mode,
such as:

• **One of abstract plan dump, abstract plan load or abstract plan replace is set**
• **statement cache and literal autoparam are set**
• **The histogram tuning factor is set to a value other than 1**

**Example 1:** Execute `sp_compatmode` with these server options:

• compatibility mode is set
dump/load/replace is "on"

statement cache is "on"

literal autoparam is "on"

histogram tuning factor is set to 20

```
1> sp_compatmodeCompatibility mode is enabled.
WARNING: Compatibility mode will not be used when 'abstract plan dump/load/replace' is on.
WARNING: Compatibility mode may not be used when statement cache and literal autoparam are enabled.
WARNING: The configuration option 'histogram tuning factor' is configured with value '20', which is not the default value in ASE 12.5. This may lead to different accuracy of statistics and different query plans.
(return status = 0)
```

Example 2: Execute sp_compatmode when compatibility mode is not set:

```
1> sp_compatmode
Compatibility mode is not enabled.
(return status = 0)
```

Note

Changing the configuration of the histogram tuning factor from the default in SAP ASE 15.0 (20) to the default in SAP ASE 12.5 (1), creates plans that are more consistent with SAP ASE 12.5.

3.5 Changes to @@qpmode Global Variable

In compatibility mode, <@@qpmode> displays the query processing mode in which the previously executed query was processed.

There are four query processing modes:

- 0 – a query that cannot be optimized; for example, create table, set, and so on.
- 1 – a query executed in full compatibility mode.
- 2 – a query executed in restricted compatibility mode.
- 3 – a query executed with the 15.0 query processor.

3.6 Diagnostic Tool

set showplan output displays the query plan in a format similar to SAP ASE 12.5.4, provided that the query is processed using full compatibility mode.
Important Disclaimers and Legal Information

Coding Samples

Any software coding and/or code lines / strings ("Code") included in this documentation are only examples and are not intended to be used in a productive system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules of certain coding. SAP does not warrant the correctness and completeness of the Code given herein, and SAP shall not be liable for errors or damages caused by the usage of the Code, unless damages were caused by SAP intentionally or by SAP's gross negligence.

Accessibility

The information contained in the SAP documentation represents SAP’s current view of accessibility criteria as of the date of publication; it is in no way intended to be a binding guideline on how to ensure accessibility of software products. SAP in particular disclaims any liability in relation to this document. This disclaimer, however, does not apply in cases of willful misconduct or gross negligence of SAP. Furthermore, this document does not result in any direct or indirect contractual obligations of SAP.

Gender-Neutral Language

As far as possible, SAP documentation is gender neutral. Depending on the context, the reader is addressed directly with “you”, or a gender-neutral noun (such as “sales person” or “working days”) is used. If when referring to members of both sexes, however, the third-person singular cannot be avoided or a gender-neutral noun does not exist, SAP reserves the right to use the masculine form of the noun and pronoun. This is to ensure that the documentation remains comprehensible.

Internet Hyperlinks

The SAP documentation may contain hyperlinks to the Internet. These hyperlinks are intended to serve as a hint about where to find related information. SAP does not warrant the availability and correctness of this related information or the ability of this information to serve a particular purpose. SAP shall not be liable for any damages caused by the use of related information unless damages have been caused by SAP’s gross negligence or willful misconduct. All links are categorized for transparency (see: http://help.sap.com/disclaimer).
go.sap.com/registration/contact.html