

# AT&T Developer Program

## Application Resource Optimizer (ARO)

### Analysis Guide

Document Created: 11/28/2011

Rev: 3

Rev Date: [7/26/2025](#)~~7/26/2025~~~~12/14/2011~~

## Legal Disclaimer

This document and the information contained herein (collectively, the "**Information**") is provided to you (both the individual receiving this document and any legal entity on behalf of which such individual is acting) ("**You**" and "**Your**") by AT&T, on behalf of itself and its affiliates ("**AT&T**") for informational purposes only. AT&T is providing the Information to You because AT&T believes the Information may be useful to You. The Information is provided to You solely on the basis that You will be responsible for making Your own assessments of the Information and are advised to verify all representations, statements and information before using or relying upon any of the Information. Although AT&T has exercised reasonable care in providing the Information to You, AT&T does not warrant the accuracy of the Information and is not responsible for any damages arising from Your use of or reliance upon the Information. You further understand and agree that AT&T in no way represents, and You in no way rely on a belief, that AT&T is providing the Information in accordance with any standard or service (routine, customary or otherwise) related to the consulting, services, hardware or software industries.

AT&T DOES NOT WARRANT THAT THE INFORMATION IS ERROR-FREE. AT&T IS PROVIDING THE INFORMATION TO YOU "AS IS" AND "WITH ALL FAULTS." AT&T DOES NOT WARRANT, BY VIRTUE OF THIS DOCUMENT, OR BY ANY COURSE OF PERFORMANCE, COURSE OF DEALING, USAGE OF TRADE OR ANY COLLATERAL DOCUMENT HEREUNDER OR OTHERWISE, AND HEREBY EXPRESSLY DISCLAIMS, ANY REPRESENTATION OR WARRANTY OF ANY KIND WITH RESPECT TO THE INFORMATION, INCLUDING, WITHOUT LIMITATION, ANY REPRESENTATION OR WARRANTY OF DESIGN, PERFORMANCE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, OR ANY REPRESENTATION OR WARRANTY THAT THE INFORMATION IS APPLICABLE TO OR INTEROPERABLE WITH ANY SYSTEM, DATA, HARDWARE OR SOFTWARE OF ANY KIND. AT&T DISCLAIMS AND IN NO EVENT SHALL BE LIABLE FOR ANY LOSSES OR DAMAGES OF ANY KIND, WHETHER DIRECT, INDIRECT, INCIDENTAL, CONSEQUENTIAL, PUNITIVE, SPECIAL OR EXEMPLARY, INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, LOSS OF GOODWILL, COVER, TORTIOUS CONDUCT OR OTHER PECUNIARY LOSS, ARISING OUT OF OR IN ANY WAY RELATED TO THE PROVISION, NON-PROVISION, USE OR NON-USE OF THE INFORMATION, EVEN IF AT&T HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH LOSSES OR DAMAGES.

## Table of Contents

1. Introduction .....	4
2. About AT&T ARO .....	5
3. Getting Started.....	6
4. Suggested Best Practices .....	7
4.1 Caching.....	9
4.1.1 Caching Diagnostics .....	10
4.1.2 Analyzing the Content.....	12
4.1.3 Conclusion .....	14
4.1.4 More cache information .....	14
4.2 Connections.....	17
4.2.1 Connection Opening .....	17
4.2.2 Multiple Simultaneous Connections.....	17
4.2.3 Periodic Transfers.....	18
4.2.4 Screen Rotation .....	20
4.2.5 Connection Closing.....	20
4.2.6 Offloading to Wi-Fi .....	21
4.3 Other .....	22
4.3.1 HTTP 1.0 Usage .....	22
4.3.2 Accessing Peripheral Applications.....	22

## 1. Introduction

The purpose of the Analysis Guide is to help you interpret the detailed results generated by the AT&T Data Analyzer component of the AT&T Application Resource Optimizer (ARO).

## 2. About AT&T ARO

AT&T ARO is a diagnostic tool for analyzing mobile web application performance that allows you to automatically profile your prototype applications to optimize their performance, make battery utilization more efficient, and reduce the network impact.

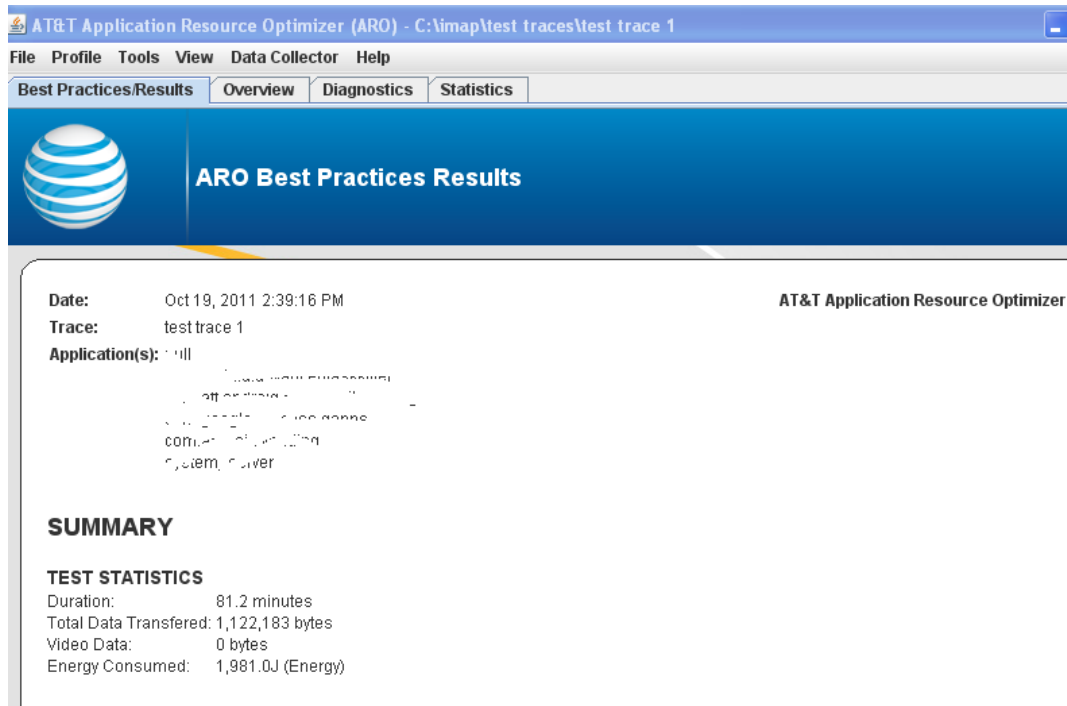
AT&T ARO has two components:

- AT&T ARO Data Collector
- AT&T ARO Data Analyzer

When using AT&T ARO, the traces run against an application by the AT&T ARO Data Collector are compared against suggested best practices tests in the AT&T ARO Data Analyzer. The Data Analyzer looks at how the application is handling caching, and how the radio and network connections are being managed in the application. By optimizing against these best practices, the application will run faster, use the network less (saving valuable battery life for users), and improve the experience of customers using the application.

This document describes the best practices tests that are run by the Data Analyzer, and shows you how to interpret the results of those tests to optimize an application.

### 3. Getting Started



**Figure 1:** The Trace Summary on the AT&T ARO Best Practices Results tab. (Please note that the application names are intentionally obscured in this figure)

There are four tabs in the AT&T ARO application. By default, AT&T ARO opens on the “Best Practices/Results” tab. When a trace file is loaded, the top section of this tab is filled in with basic information, including a list of applications that were running when the trace data was captured by the AT&T ARO Data Collector. This Trace Summary provides a basic scorecard of test statistics, and shows pass/fail results for each of the best practices that the trace is evaluated against.

In the following sections, the Analysis Guide introduces the Best Practices tests, and provides examples of how to interpret the results for each of the test categories:













## 4. Suggested Best Practices

The best place to start analyzing the trace data is to review the results of the suggested best practices tests on the Best Practices/Results tab. The tab provides a rough guide to issues that can be resolved.

On this tab, the results are compared to best practices and references for improved optimization. The trace is tested against 10 suggested best practices automatically with 2 additional self tests. The suggested best practices tests are broken into 3 types: Caching, Connections, and Other.

The following figure shows the 12 best practices tests:

## TESTS CONDUCTED

-  Caching: Duplicate Content
-  Caching: Cache Control
-  Caching: Content Expiration
-  Caching: Content Pre-fetching
-  Connections: Connection Opening
-  Connections: Unnecessary Connections - Multiple Simultaneous Connections
-  Connections: Inefficient Connections - Periodic Transfers
-  Connections: Inefficient Connections - Screen Rotation
-  Connections: Inefficient Connections - Connection Closing Problems
-  Connections: Inefficient Connections - Offloading to WiFi when Possible
-  Other: Accessing Peripheral Applications
-  Other: HTTP 1.0 Usage

**Figure 2:** Test results of the 12 best practices tests. Green check means pass, red “x” means fail, and the blue image indicates a self test (A test that is not automatically run by the Data Analyzer).

Best practices are organized by type, and there is a section header for each type on the Best Practices/Results tab. In the section header for each type, the results are graded as either a Pass or Fail. A Pass is given to the group of tests within that type if all of the



tests in the type have passed (see figure 3 that follows). The group of tests in that type fails if there is just one best practice test of that type that has failed (see figure 4 that follows).

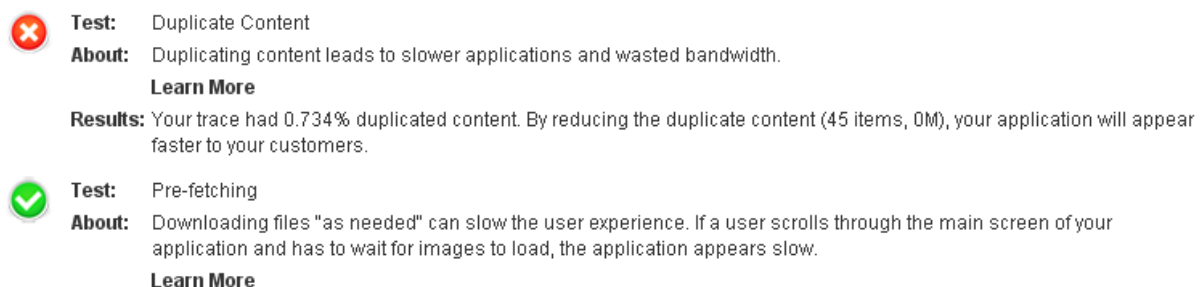


**Figure 3:** Pass – If all of the best practice tests in the section pass, the whole section passes.



**Figure 4:** Fail – If one best practice test in the section fails, the whole section fails.

Inside each section, there are more details about each test and specific information on the trace. The text “Learn More” links to the AT&T Developer Program website for details about the issue and several proposed remedies. By clicking the title of the test, the AT&T Data Analyzer displays another section that provides even more details about the issue.



**Figure 5:** Detail on two best practices tests with specific information about the trace. In this example, the Pre-fetching test has passed, but the Duplicate Content test has failed because 45 small items were downloaded multiple times causing 0.734% of the total bytes of content to be duplicated.

The following sections take a deeper look into analyzing each of the three categories of best practices.

## 4.1 Caching

The AT&T Data Analyzer presents the results of the caching best practices tests in the Overview, Diagnostics, and Statistics tabs. This example shows you how to analyze the

caching information by following the analysis through the layers of detail presented in these tabs.

The top level of detail is displayed in the Overview tab, where the Duplicate Content table lists all of the duplicate content that was observed in the trace. The table displays the content type indicated by a label, the time at which the content was downloaded, the file name and path which indicates the file location on the server, and the file size in bytes. By default, this table is sorted by the time, in seconds, at which the content was downloaded.

Duplicate Content				View	Save As...
Duplicate Content Type	Time	File Name	File Size (bytes)		
ORIGINAL_FILE	111.1	/handset/	18,513		
OBJDUP_NOT_EXPIRED	279.1	/handset/	18,513		
OBJDUP_NOT_EXPIRED	329.9	/handset/	18,513		
OBJDUP_NOT_EXPIRED	390.4	/handset/	18,513		
OBJDUP_NOT_EXPIRED	444.4	/handset/	18,513		
OBJDUP_NOT_EXPIRED	486.5	/handset/	18,513		

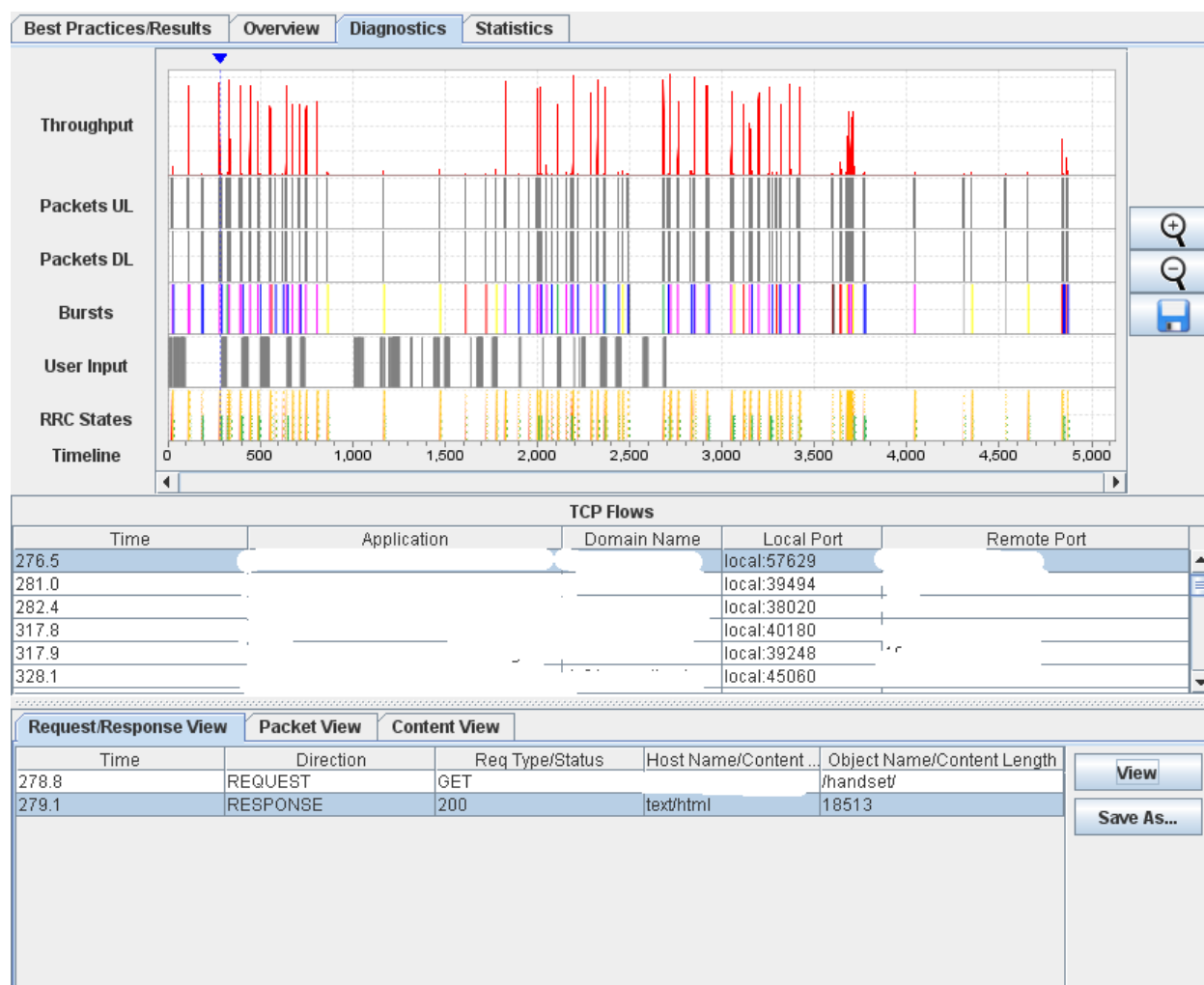
**Figure 6:** The Duplicate Content table in the Overview tab, sorted by time.

In the Duplicate Content table shown in figure 6, a file was first downloaded at 111.1 seconds. The fact that it was the first download of the file is indicated by the content type label ORIGINAL\_FILE. Following down the rows of data, the table shows that the same file was subsequently downloaded at 279.1 seconds, 329.9 seconds, and at least three additional times. The content type label “OBJDUP\_NOT\_EXPIRED” indicates that the file was a duplicate that was not expired.

The Duplicate Content table can be sorted on any column by clicking the column headers. This provides many useful views into the information. For example, if there are multiple duplicate files, sorting on the File Name column shows how many times each file was downloaded. To export the data from the Duplicate Content table in csv format, right-click on the table and click “Export”.

### 4.1.1 Caching Diagnostics

Double-clicking on a file in the Duplicate Content table opens the Diagnostics tab which displays a deeper layer of detail about the flow of content that occurred during the trace. The following figure shows the Diagnostics Tab after the second row in the Duplicate Content table (figure 6), containing data for the duplicate file that occurred at 279.1 seconds in the trace timeline, is double-clicked.



**Figure 7:** The Diagnostics Tab. (Please note that the Application, Domain Name, and Remote Port data in this figure have been intentionally obscured).

The information in the Diagnostics tab is oriented to a position in the trace timeline indicated by the blue carat at the top of graph. The tables below the graph have highlighted the rows of data that occurred at that same position on the timeline.

The top graph in the Diagnostics tab is a representation of the flow of content in the trace that was captured by the AT&T Data Collector, and is the best place to begin identifying and analyzing network traffic issues. The default view of the graph (as shown in figure 7) displays the throughput, all of the packets uploaded (UL) and downloaded (DL), the bursts (groupings of packets that have been assigned various categories), the user inputs, and the RRC state machine model states from the trace. The graph can be

configured, and the fields in it can be added or removed, by selecting “Options” in the “View” menu (visible in figure 1) of the AT&T Data Analyzer.

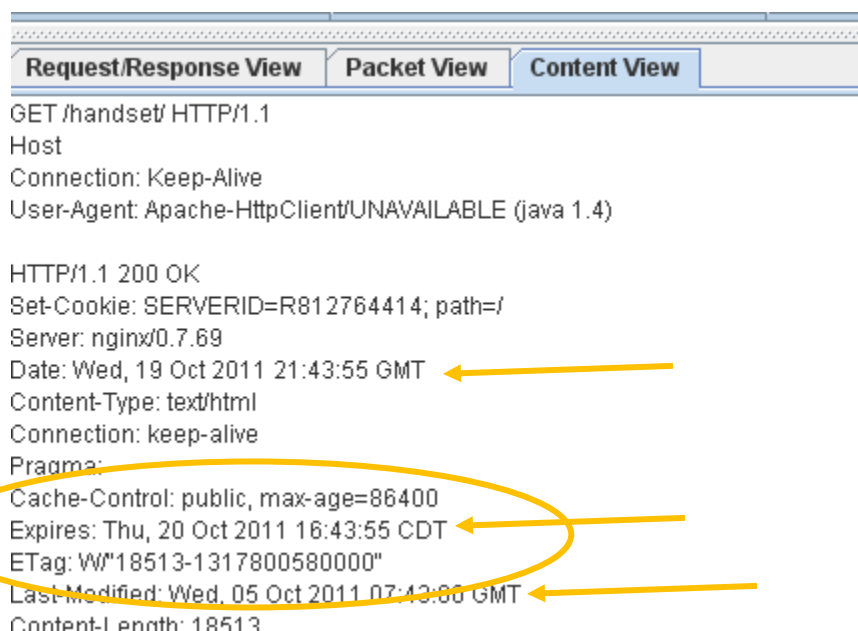
The TCP Flows table in the Diagnostics tab is a list of all TCP connections. By default, this table is sorted by the time, in seconds, at which the connections occurred. The Application column names the application that initiated the TCP flow, and the Domain Name is where the connection goes to from the phone. The Local Port is essentially the local IP address, while the Remote Port is the server IP address.

The bottom table in the Diagnostics tab has three different views into the TCP flow data that is highlighted in the TCP Flows table. By default, this table shows the Request/Response view which gives a breakdown of all the requests/responses for the highlighted TCP flow. Clicking on the “Packets View” tab displays the packets involved in this TCP flow, and the “Content View” tab displays the content. By highlighting a line in the table and clicking the “View” button to the right of the table, you can view the contents that were sent during the trace. To save the contents to a file, click the “Save As” button.

#### **4.1.2 Analyzing the Content**

In the example in figure 7, the blue carat at the top of the graph indicates that the information on the Diagnostics Tab is oriented around the trace timeline position of 279 seconds. This is the position at which the duplicate content occurred (in figure 7). The TCP Flows table, has the row highlighted that shows the TCP flow containing the duplicate content. The bottom table (in the Request/Response view) shows that there was a REQUEST for this file (the host name has been removed from the figure to anonymize the trace), and a subsequent 200 RESPONSE from the server that delivered the file to the application.

Clicking on the “Content View” tab provides one further layer of detail to help analyze what caused the duplicate transfer of content. The following figure shows the “Content View” of the 200 RESPONSE from the server that sent the duplicate data at 279.1 seconds in the trace.



**Figure 8:** The Content View of the TCP Flow Table in the Diagnostics Tab.

In the “Content View” of the response shown in figure 8, the circled header was inserted by the server, and identifies a “max-age” indicating that the file can be cached for 86400 seconds (24 hours). The “Date:” (pointed at by the first arrow) shows that the file was served up on 10/19/2011 at 21:43 GMT, and that it “Expires:” (pointed at by the second arrow) on 10/20/2011 at 16:43 CDT. The “Expires:” time translates to 21:43 GMT, which is exactly 24 hours after the download, and matches the amount specified by “max-age”. The third arrow points to the header line showing that the file was “Last Modified:” on 10/5/2011.

To implement a cache properly on this application, this file should be downloaded once on 10/19, and each subsequent request on 10/19 should be served from the cache. After the expiration date on 10/20, the application should ask the server if the file has been modified. If it has not, the server should tell the application (via a 304 Not Modified response) to use the previously cached version and the 24 hour cache timer will restart.

The analysis of this header also indicates that this file does not appear to be changed very often. In this case, the application developer should consider increasing the “max-age” time to every two days, or even once a week. This would reduce the number of pings (requests sent to the server).

### **4.1.3 Conclusion**

The analysis of this caching information used the AT&T Data Analyzer to work through the layers of caching information in the application trace. It started at a higher level in the Overview tab which showed that there was duplicate content being sent. By clicking on the content, it moved to the Diagnostics tab and identified the TCP flow and the server requests and responses for the duplicate file. Finally, by looking at the Content View of the TCP flow, the cache header for the file could be analyzed.

Based on this analysis, we're able to conclude that there is no local cache set up for this application. The proposed fix is to add a local cache on the device. If the Content View had not shown a cache header like the one circled in figure 8, an additional fix would be to add a cache header on the server.

### **4.1.4 More cache information**

The AT&T Data Analyzer has another tab of information that can be used to analyze the caching issue in this example even further. By clicking on the Statistics tab and scrolling down about 2/3 of the screen, the HTTP Cache Statistics are displayed as in the following figure.

Best Practices/Results Overview Diagnostics <b>Statistics</b>			
<b>HTTP Cache Statistics</b>			
		% of Responses	% of Bytes
----- Cacheable vs. non-cacheable -----			
Cacheable:	49.6	93.4	
Specified - no cache:	50.4	6.6	
----- Cache simulation results -----			
Acceptable Behavior			
Files downloaded once:	9.7	18.8	
Files specified as "No-Cache":	50.4	6.6	
Expired, but correct 304 response sent from server:	0.0	0.0	
Expired, downloaded again, but file has changed:	0.0	0.0	
Duplicate File Download			
Duplicate download (not expired):	39.8	74.6	
Duplicate download (expired, but no "If-Modified-Since" header sent):	0.0	0.0	
Duplicate download (expired, but "If-Modified-Since" header ignored):	0.0	0.0	
Duplicate File Download: Streaming			
Partial duplicate download (Not Expired):	0.0	0.0	
Partial duplicate download (expired, but no "If-Modified-Since" header sent):	0.0	0.0	
Partial duplicate download (expired, but "If-Modified-Since" header ignored):	0.0	0.0	
----- Duplicate File Analysis -----			
Duplicate download (Cache not expired):	73.3	99.3	
Duplicate download (24 hr cache not expired):	26.7	0.7	
Duplicate download (Cache expired):	0.0	0.0	
Duplicate download (24 hr cache expired):	0.0	0.0	

**Figure 9:** The HTTP Cache Statistics section of the Statistics tab.

The statistics provided in the various sections of HTTP Cache Statistics gives the following information about this example application.

The first section, "Cacheable vs. non-cacheable", shows how many of the files were cacheable versus the number of files that had no specific cache headers. The second line, "Specified – no cache", shows that in this example 50.4% of files had no cache header (meaning that they could not be cached), and that these files accounted for 6.6% of the total data traffic in bytes.



The second section, “Cache simulation results” measures how the application performed acceptable caching behavior. These statistics indicate the following about this example.

These types of acceptable caching behavior graded well statistically:

1. 9.7% of the files were downloaded once.
2. 50.4% of the files were not allowed to be cached.
3. 0% of files were cached – expired, with the server sending the correct 304 “not modified” response.
4. 0% of the files were expired and out of date – and then downloaded again.

While the following lines indicate that these types of acceptable caching behavior did not grade well :

1. In the line item for “Duplicate download (not expired)”, 39.8% of the downloads (accounting for 74.6% of the total bytes) was duplicate content.
2. 0% of the streaming files were duplicate. This occurs when specific byte ranges of videos are sent down. It is possible that certain byte ranges are sent in a duplicate manner.

In the W3C specification, there is an assumption that all files without cache headers should be cached for 24 hours. The “Duplicate Files Analysis” section of “HTTP Cache Statistics” breaks down files with defined cache times and those with the 24 hour cache expiration.

In this example, the statistics show that 73.3% of the duplicate files had correct headers, and that 26.7% had no headers but fall into the 24 hour cache requirement. The proposed fix for this is to use the information on the “Overview” tab to find the duplicate files with no cache headers, and add the headers on the server. The “Duplicate download – 24 hr cache not expired” line shows that this only accounts for .7% of all the byte traffic in the application, but the fix would make for more efficient coding.

The main conclusion from the HTTP Cache Statistics table is that 74% of all data sent was duplicate content that did not require resending. By instituting a cache, the server load will drop by ~75%, the network traffic will decrease, and users will likely see a faster application.



## 4.2 Connections

In the previous example - analyzing the results of the Caching best practices tests - it is relatively easy to identify the wasteful behavior of downloading the same file over and over again. The second groups of best practices test results –Connections – are presented in the same layers of detail in the Overview, Diagnostics, and Statistics tabs, but the results require deeper analysis to understand how identifying unnecessary and inefficient connections can speed up an application.

The following examples analyze the results of the Connections best practices tests, and show that by efficiently organizing the radio connections made by an application, large application performance improvements can be achieved.

### 4.2.1 Connection Opening

The Connection Opening test is a self test. This means that it is not conducted automatically by the Data Analyzer, but static information about connection opening is displayed in the results. Here are some important things to look for when analyzing the Connection Opening information.

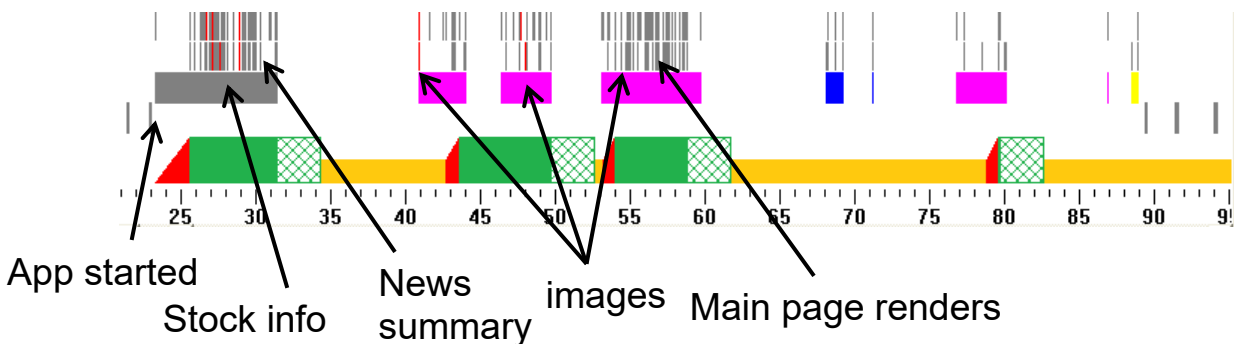
In the Diagnostics tab, Application-initiated bursts are colored in red. If there are many of these type of bursts, it indicates that the application is pinging the network without user input. If this is the case, it would be more efficient to download more data per connection in order to decrease the quantity of these connections.

Usability studies can also be helpful in analyzing connection opening. For example, if a usability study discovered that 80% of users are reading the first story in an application. It would make sense to “prefetch” the images in that story whenever the application starts, in order to speed up the loading of the story.

### 4.2.2 Multiple Simultaneous Connections

Managing the way in which files are downloaded can greatly speed up the way an application runs. The following example analyzes results from the Unnecessary Connections - Multiple Simultaneous Connections test, and shows how grouping connections can speed up an application.

The following figure is taken from the Trace chart in the Diagnostics tab. This part of the trace shows an application starting up:



**Figure 10:** An application trace showing an application at start-up.

In this figure, the radio is open for 10 seconds between the download of the text files (Stock info and News summary) and the images, and there are two gaps in the image downloads. If these connections all occurred simultaneously, or immediately back to back, the application would load faster, and users would see the content faster.

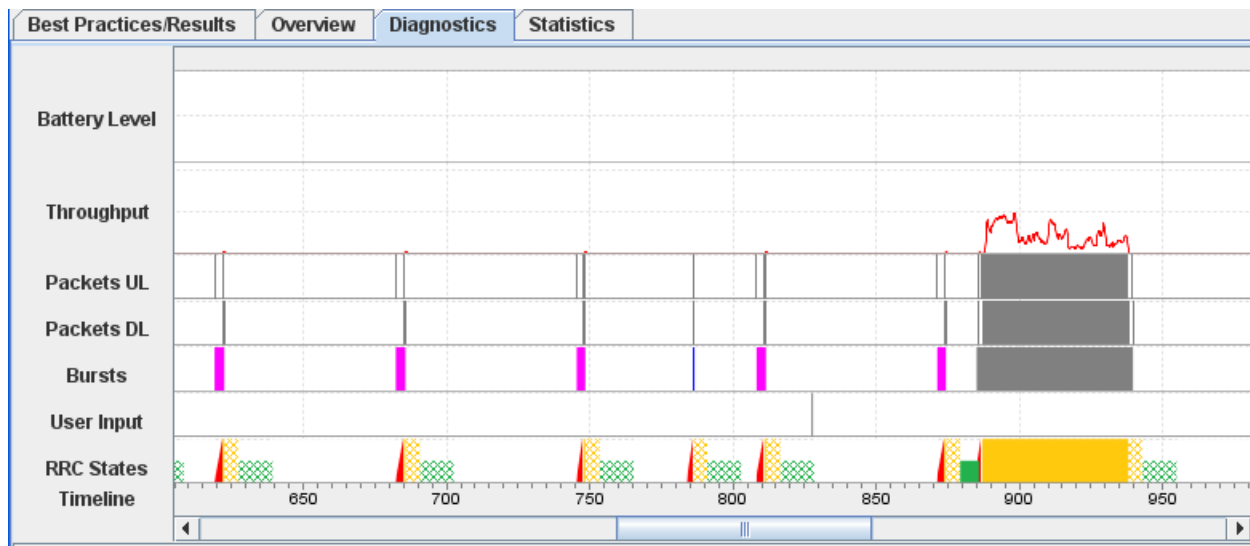
By eliminating the space between these bursts, the main page of this application would render 15 seconds faster. A dramatic 50% improvement!

This same fix would also save battery usage, as the radio would be off much sooner.

### 4.2.3 Periodic Transfers

There are several reasons for periodic transfers (pinging) in an application such as; Analytics, Keep-Alive messages, and Advertising. However, the problem with constant pinging is that the RRC state machine keeps the radio connection open for a long period after the short periodic transfer. This causes fast drain of the battery, and prevents users from using an application for a longer period of time. For more details on the RRC state machine, please read our [developer best practices](#).

The following example analyzes results from the Inefficient Connections - Periodic Transfers test In the Data Analyzer to show how different types of periodic transfers can be optimized in an application. In the following figure, the Trace chart in the Diagnostics tab shows a ping every 60 seconds.



**Figure 11:** The Trace chart in the Diagnostics tab showing the graphical representation of an internet radio application trace.

The trace data in figure 11 is from an internet radio application. The periodic transfers (colored purple/pink in the Trace chart), show that the application is taking audience measurements every 60 seconds. The gray colored bar in the Trace chart (a Large Burst) shows where a song is being downloaded. By analyzing these two periodic transfers in the Trace chart, you can see that the amount of time spent with the radio on for audience measurement analytics is close to the same amount of time spent downloading music.

The following figure showing the Burst Analysis table in the Statistics tab identifies the same correlation:

Best Practices/Results	Overview	Diagnostics	Statistics				
Burst Analysis							
Burst	Bytes	% of Bytes	Energy	% of Energy	DCH (Active)	% DCH (Active)	JpKB
TcpControl	0	0.0	22.5	5.8	14.2	4.1	
UserInput	62,756	0.6	17.6	4.5	14.8	4.3	0.0
App	110,721	1.1	39.0	10.0	35.4	10.3	0.0
LargeBurst	9,580,705	98.1	147.8	38.0	179.2	52.4	0.0
Periodical	12,088	0.1	162.0	41.6	98.7	28.8	1.7

**Figure 12:** The Burst Analysis table in the Statistics tab showing Burst statistics.

Compare the “Large Burst” and “Periodical” rows in the Burst Analysis table (figure 12). The “Large Bursts” (songs), account for 98.1% of the data (bytes), but only 38% of the energy consumption. While the periodic transfers (the audience measurement analytics) account for 0.1% of the data, but consume 41.6% of the total radio energy used.

Other common uses of periodic transfers include Keep-Alive messages and Polling.

Keep-Alive messages are meant to keep persistent TCP connections alive. However, these messages are unnecessary if they are much more frequent than the timeout for the TCP connection. For instance, AT&T’s Firewall Keep-Alive is 30 minutes, and the shortest example found among major carriers is 4 minutes, but applications often send Keep-Alive pings at a much faster rate.

Polling is done by an application to check the server for new messages at a regular interval. From an energy perspective, it is better to keep a static TCP connection open and allow messages to be pushed to the application. For instance, AT&T keeps TCP connections persistent for 30 minutes, so polling that is much more frequent than that is unnecessary.

#### **4.2.4 Screen Rotation**

The Screen Rotation test is a self-test. This means that the test is not conducted automatically by the AT&T Data Analyzer, but static information about screen rotation is displayed in the results. When capturing a trace using the AT&T Data Collector, the best way to conduct this test is to rotate the screen of the device during the trace, and then use the Data Analyzer to determine if the application contacted the server during the rotation. In some applications, content is completely reloaded for the new screen orientation.

#### **4.2.5 Connection Closing**

The results of the Connection Closing test can identify another common application issue.

The best place to begin analyzing information on connection closing is in the Overview tab, at the bottom of this tab, the “Accessed Domains” and “Domain TCP Sessions” tables provide a lot of insight into this issue. The “Accessed Domains” table lists all of the domains accessed during the trace. It shows the number of sessions, the average session length, and the number of files downloaded from that domain. By clicking on a domain with a long average session length, all of the sessions from that domain will be displayed in the “Domain TCP Sessions” table on the right.

In the following example, the domain in the last row of the “Accessed Domains” table, with 106 accesses, has been clicked, and the first 9 sessions are displayed in the “Domain TCP Sessions” table on the right:

Accessed Domains				Domain TCP Sessions					
Domain name	TCP...	Avg. session length ...	Files downloaded	Time	Remote IP Address	Local Port	Session ...	Session ...	Closed By
...	...	14118.4	0	21.8	...	54399	5.4	2.3	Client
...	...	112.7	1	108.0	...	41993	6.1	2.1	Client
...	...	13.7	1	113.9	...	44485	164.9	67.3	Server
...	...	14199.5	1	276.5	...	57629	4.7	1.0	Client
...	...	12.4	0	281.0	...	39494	1.9	1.0	Client
...	...	21040.9	2	282.4	...	38020	8.2	6.8	Client
...	...	24.2	0	317.9	...	39248	242.3	4.1	Client
...	...	88.0	5	328.1	...	45060	4.2	1.4	Client
...	...	10666.4	103	331.9	...	45110	58.6	57.6	Client

**Figure 13:** The Accessed Domains and Domain TCP Sessions tables at the bottom of the Overview tab.

The “Domain TCP Sessions” table (on the right side in figure 13) contains the following columns from left to right: The time of the session (Time), the connection IPs on the remote server (Remote IP Address), the local IP port (Local Port), the session length (Session Length), the time between the last packet and the session close request (Session Close), and what side of the connection closed the connection (Closed By).

In this example, the data in the Session Close column, shows that many of the connections closed within a few seconds of the last packet, but that two were nearly a minute later (67.3 and 57.6 seconds). By double-clicking on one of these TCP sessions in the table, the Diagnosis tab will open, where the files and packets associated with the TCP connection are displayed. By analyzing this information along with the server data, it can be determined why these connections did not close as promptly as the others.

On the Statistics tab, the amount of energy consumed by the radio for these connections is listed as “TCPControl” in the Burst Analysis table. See figure 12 for an example of the Burst Analysis table in the Statistics tab. The TCPControl bursts are displayed in the first row of the table.

## 4.2.6 Offloading to Wi-Fi

The Offloading to Wi-Fi when Possible test analyzes how many large bursts of data occurred in the trace. The number of large bursts can be seen in the Trace Chart in the Diagnostics tab and in the Burst Analysis table in the Statistics tab. The results of this test are mostly informational, but the purpose is to make you aware that switching to Wi-Fi will speed up an application because there is no TCP connection delay or state machine when running over Wi-Fi. A good solution to this issue is to add a dialog to the startup of the application that offers the user the opportunity to switch to Wi-Fi.

## **4.3 Other**

The Other category of best practices tests In the AT&T Data Analyzer contains the Accessing Peripheral Applications and HTTP 1.0 Usage tests.

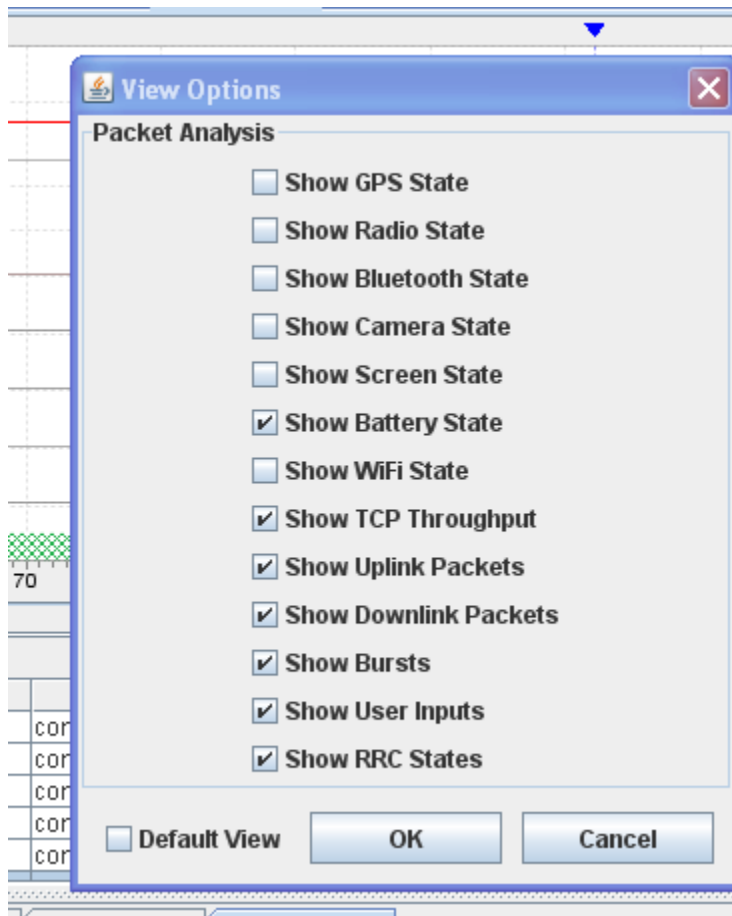
### **4.3.1 HTTP 1.0 Usage**

The HTTP 1.0 Usage test looks for HTTP 1.0 in the header of the loaded trace files, if it is found, the test results indicate that using HTTP 1.1 instead, allows multiple items to be downloaded per connection, which is more efficient for an application.

### **4.3.2 Accessing Peripheral Applications**

Selecting “Options” in the “View” menu of the AT&T Data Analyzer opens the “View Options” dialog (visible in figure 1). In the “Packet Analysis” section of the “View Options” dialog, you can select which peripheral applications will be included in the Trace chart at the top of the Diagnostics tab (The Trace chart is shown in figure 11.).

The following figure shows the “View Options” dialog.



**Figure 14:** The Packet Analysis section of the View Options dialog.

Adding a peripheral application like GPS, Bluetooth, or camera to the Trace chart shows how and when these peripheral applications were on during the trace.

For example, a common situation occurs where the GPS does not acquire a location, but also does not time out after a few minutes. This means that the GPS ends up running in active mode for as long as the application does – draining the battery of the user for no apparent reason. Charting the GPS State in the Trace chart can identify this problem in an application.

To correct this situation, the best practice is to build in a timeout into all peripheral application use, so that a peripheral application turns off when it is not in use (or is not working). In the case of GPS, a fall back check could be set up to occur X minutes after the initial attempt to acquire a location, to see if a location is available at that moment.

Another way to streamline GPS usage is to determine what type of location data is important to the application. Does the application need to show a very specific location (such as the bus stop at 3<sup>rd</sup> and Pine St. in Seattle, WA)? Or is network location alone sufficient? It is faster (and uses less energy) to use the network location if knowing a very specific location is not necessary.