webtrends™

Webtrends Analytics On Demand Implementation Guide

Spring 2011 Edition |

Disclaimer

This document and the software, subscription service and/or technology described in this document are furnished under and are subject to the terms of a separate license agreement, a subscription service agreement or a services agreement.

EXCEPT AS EXPRESSLY SET FORTH IN A LICENSE AGREEMENT, SUBSCRIPTION SERVICE AGREEMENT OR A SERVICES AGREEMENT, WEBTRENDS INC. PROVIDES THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. NO RIGHTS OF ANY KIND ARE GRANTED TO YOU IN THE SOFTWARE, SUBSCRIPTION SERVICE AND/OR TECHNOLOGY DESCRIBED IN THIS DOCUMENT, UNLESS SUCH RIGHT HAS BEEN GRANTED TO YOU UNDER A SEPARATE LICENSE AGREEMENT, SUBSCRIPTION SERVICE AGREEMENT OR SERICES AGREEMENT.

You agree that you shall not loan, sell, or otherwise transfer this document.

Except as expressly set forth in a license agreement, subscription service agreement or services agreement, you agree that you shall not reproduce, store in a retrieval system, provide access to or transmit in any form or by any means, electronic, mechanical, or otherwise, all or any part of this document or the software, subscription service and/or technology described in this document. Some companies, names, and data in this document are used for illustration purposes and do not represent real companies, individuals, or data.

This document may include technical inaccuracies or typographical errors. Webtrends Inc. may make improvements in or changes to the software, subscription service and/or technology described in this document at any time without notice.

U.S. Government Restricted Rights: The software is "commercial software." If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government's rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement. Government technical data and software rights related to the service include only those rights customarily provided to the public as defined in this Agreement. This customary commercial license is provided in accordance with FAR 12.211 (Technical Data) and FAR 12.212 (Software) and, for Department of Defense transactions, DFAR 252.227-7015 (Technical Data - Commercial Items) and DFAR 227.7202-3 (Rights in Commercial Computer Software or Computer Software Documentation).

Trademarks

Webtrends, Webtrends logo, Webtrends Visitor Intelligence, Webtrends Visitor Intelligence logo, Score, Score logo, Analytics, Analytics logo, Ad Director, Ad Director logo, Marketing Warehouse, Marketing Warehouse logo, Visitor Data Mart, Visitor Data Mart logo, Explore, Explore logo, Optimize, Optimize logo, Social Measurement, Social Measurement logo, Segments, Segments logo, and Open Exchange are trademarks or registered trademarks of Webtrends Inc. or its subsidiaries in the United States and other jurisdictions. All other company and product names may be trademarks or registered trademarks of their respective companies.

Contact Information

Sales and General	Support and Services	Online Resources
Webtrends, Inc. 851 SW 6th Ave. Suite 1600 Portland OR 97204 Phone: 1-503-294-7025 Fax: 1-503-294-7130 US Toll Free: 1-877-WEBTRENDS (1-877-932-8736) **Email:** feedback@webtrends.com	**Direct Technical Support:** North America and Caribbean: 1-503-223-3023 Central and South America: 1-503-223-3023 Asia Pacific, Australia, New Zealand: 1-503-223-3023 Europe, Middle East, Africa: +44 (0) 1784-463-555	**Webtrends Home Page:** http://www.webtrends.com **Webtrends Support Center:** http://www.webtrends.com/support **Webtrends Professional Services (Consulting and Training):** http://www.webtrends.com/services **Online Customer Center:** http://www.webtrends.com/resources

About This Book and the Library

This guide is intended to help the Webtrends administrator implement and configure Webtrends Analytics for reporting. Webtrends Analytics On Demand offers a variety of powerful analysis and reporting options. The features and reports available to you are dependent on how your company is licensed to use Webtrends. If you have questions about the Webtrends features you are licensed to use, please see your Webtrends administrator.

The Webtrends Implementation Guide provides in-depth configuration and reference information for Webtrends administrators. It includes:

- Detailed information for tagging your web site for data collection
- A complete reference to Webtrends query parameters
- Best practices for reporting on unique visitors
- A discussion of Webtrends On Demand and security

Intended Audience

This book provides information for administrators who are responsible for setting up Webtrends Analytics On Demand.

Other Information in the Library

The library provides the following information resources:

Help
Provides context-sensitive information and step-by-step guidance for common tasks, as well as definitions for each field on each window.

Webtrends Administration User's Guide
This guide provides complete information for using Webtrends Administration to set up and customize core operations such as data collection, analysis, report content and style, and visitor session tracking. It includes conceptual and procedural information about features such as custom reports, data filtering, scenario analysis, and Express Analysis; assistance with common administrative concerns such as job scheduling and table limiting; and reference information such as the Webtrends Query Parameter Reference.

Webtrends Analytics Reports User's Guide
This guide provides users who primarily use Webtrends Analytics Reports with the information they need to navigate, customize, save, and export reports and report data.

Note
Users who only have View Reports permissions automatically use Webtrends Analytics Reports instead of Webtrends Administration. While they can view the reports, they may not have access to any of the other controls. We recommend distributing the *Webtrends Analytics Reports User's Guide* to these users as a introduction to navigating Webtrends reports and report data.

Webtrends Marketing Warehouse Software User's Guide

This guide includes information about using Webtrends Marketing Warehouse for ad hoc data analysis, using Webtrends Explore to analyze web business events by segment, and using Webtrends Score to identify qualified users based on their web site actions. Webtrends administrators can also find information about installing, implementing and using Webtrends Marketing Warehouse.

Webtrends Visitor Data Mart On Demand User's Guide (Formerly Webtrends Marketing Warehouse On Demand User's Guide)

This guide includes information about using Webtrends Visitor Intelligence for ad hoc data analysis, using Webtrends Explore to analyze web business events by segment, and using Webtrends Score to identify qualified users based on their web site actions. It also provides a detailed reference to Visitor Intelligence report data. Webtrends administrators can also find information about implementing and using Webtrends Marketing Warehouse.

Webtrends Analytics Software Implementation and Maintenance Guide

A step-by-step guide for administrators who are responsible for installing, setting up and maintaining Webtrends Analytics Software. It includes information about licensing, JavaScript tagging, profile setup, security, cookie implementation, performance tuning, and system backups. It also includes the Webtrends Query Parameter Reference.

Webtrends Analytics On Demand Implementation Guide

A step-by-step guide for administrators who are responsible for implementing Webtrends Analytics On Demand. It includes information about licensing, JavaScript tagging, and profile setup. It also includes the Webtrends Query Parameter Reference.

Webtrends SmartSource Data Collector User's Guide

This guide provides instructions for installing, configuring, and maintaining Webtrends SmartSource Data Collector, including information about client- and server-side JavaScript tags and cookie tracking.

Webtrends SmartView User's Guide

A guide to installing and using SmartView and configuring Webtrends to work effectively with SmartView reporting.

Webtrends SmartReports User's Guide

A guide to using Webtrends SmartReports with Webtrends Analytics reporting for powerful data integration and analysis in the Microsoft Excel environment.

Webtrends Visitor Data Mart Schema Reference (formerly Webtrends Marketing Warehouse Schema Reference)

Provides an overview of the Marketing Warehouse databases for experienced database administrators. This guide helps you understand the data in the Marketing Warehouse, giving you the foundation you need to use the data productively. It provides instructions for populating the Marketing Warehouse databases using Webtrends Administration and for viewing the data once it is available. It also describes how the databases are constructed and how that affects the function of the different types of data.

Webtrends Programmer's Reference

This guide provides conceptual, procedural, and referential information that allows experienced programmers to customize Webtrends data collection and reporting. It provides instructions for using the Webtrends ODBC Driver to query both the Marketing Warehouse and the Webtrends Analytics Report databases. It also includes documentation for the Active X, C, and Post Plug-Ins that can communicate with Webtrends Analytics.

Webtrends Guide to Web Analytics

This guide provides an introductory conceptual overview of web analytics, supplemented with examples, graphics, and practical worksheets to help you understand Webtrends architecture and create a strategy for customizing Webtrends Analytics for your key business metrics. Topics covered in this guide include collecting web activity data, understanding visitor behavior, filtering and analyzing your data, measuring acquisition, conversion, and retention, and integrating web analytics data with other business data.

Providing Feedback

Your comments are very important to us. Please take the time to let us know about your Webtrends experience by doing one of the following:

- Click **Customer Center** in the upper right corner of the Webtrends banner. Then click **Contact Us** and click **Submit Product Feedback** in the right pane.
- From Webtrends Analytics Reports, click **Help > Feedback** from the upper right corner of the report.

The Feedback page of the Webtrends web site opens in a new browser window. You can use it to report a bug, request a feature, or give general feedback about your user experience.

Documentation Center

The Webtrends Documentation Center brings together a variety of materials and references to help you learn to use Webtrends products more effectively. To access the Documentation Center, go to webtrends.com, click Support, then click Product Documentation (under Documentation and Downloads).

Conventions

The library uses consistent conventions to help you identify items throughout the documentation. The following table summarizes these conventions.

Convention	Use
Bold	• Window and menu items • Technical terms, when introduced
Italics	• Book and CD-ROM titles • Variable names and values • Emphasized words
`Lucida Console`	• File and folder names • Commands and code examples • Text you must type • Text (output) displayed in the command-line interface

Table of Contents

Server-Side Integration for Webtrends Analytics On Demand

Chapter 4

Customizing JavaScript Tags

Chapter 5
Tracking Complex Web Page Interactions Using dcsMulti-Track

Chapter 6
Creating Webtrends Analytics Profiles

Chapter 7
Setting Up Webtrends Users and Roles

Chapter 8
Tracking Visitor Sessions

Chapter 9
Webtrends Query Parameter Reference

Chapter 10
Securing Your Implementation

Chapter 1

Licensing

This chapter provides information about how Webtrends is licensed and tells you how to check the status of your license.

How Webtrends On Demand Is Licensed

Your license determines:

- How long you can use Webtrends On Demand.
- Which core Webtrends products you can install and use. Core products include Webtrends Analytics, Webtrends Marketing Warehouse, Visitor Intelligence, Score, and Ad Director.
- Which *report packs* you can use. Report packs are collections of reports that are grouped according to purpose. For example, the Commerce Report Pack includes reports about products and purchase-related visitor segments. Report packs allow you to report on the type of information you need and give you the flexibility to add report packs as your needs grow.
- Which product add-ons, such as Advanced SmartView and Custom Reporting, you can use.
- The number of server calls that Webtrends On Demand collects during your licensing period. If you reach the server call limit, you can contact your Account Manager to purchase additional server calls. For more information, see "Understanding Server Call Licensing" on page 1.
- How many Webtrends Analytics profiles you can create. For more information about licensing Webtrends Analytics parent-child profiles, see "Understanding Profile Licensing" on page 2. Your license does not limit the number of Marketing Warehouse profiles.
- The number of Webtrends Analytics reports you can export. If you reach the limit for report exports, contact your Account Manager to purchase additional exports.
- How many events you can collect and store in the Marketing Warehouse. If you reach the limit for events collected or stored, contact your Account Manager to purchase additional events.

Understanding Server Call Licensing

Webtrends Analytics On Demand is licensed according to server calls analyzed per licensing period. For example, during a 12 month period, your Webtrends Analytics activation may be licensed for up to 20 million server calls.

In a licensing context, a *server call* is defined as a file with a web page extension (such htm, html, asp) that is requested by a web site visitor to one of the domains being analyzed. However, for Webtrends On Demand, all hits (or files) regardless of the extension are considered server calls.

If Webtrends Analytics analyzes data that is processed at a later date by a different profile or by the same profile, the server call is not counted again. Server calls are counted before applying any data filters. Although multiple frames on one web page are counted as separate (multiple) server calls, you can avoid this problem by excluding the Webtrends JavaScript tag from the secondary frames.

Understanding Profile Licensing

Your license determines the number of profiles you can create. You can distribute the total number of profiles among your *child accounts*. Child accounts are sub-accounts within your main license.

Child accounts are counted differently toward your license total depending on whether you use Basic Analysis or Full-Featured Analysis to create Parent-Child profiles.

A Basic Analysis Parent-Child profile with up to 10 child profiles counts as one profile. Each group of 10 additional child profiles counts as one additional licensed profile. For example, a Parent-Child profile using Basic Analysis with between 11 and 20 children counts as 2 profiles. A Parent-Child profile using Basic Analysis with between 21 and 30 children counts as 3 licensed profiles, and so on.

For a Full-Featured Analysis Parent-Child profile, each child profile counts as one profile. For example, a Parent-Child profile with 3 children counts as 3 licensed profiles.

Checking the License Status

The **Edit Account** dialog box in Webtrends Accounts allows you to view the status of your profiles, custom reports, and report export limits.

To check your license status:

1. From Webtrends Analytics Accounts, select **Account > Edit Account** in the left pane.
2. Click **Settings**.

Checking Your Server Call and Event Use

You can view the number of server calls and events collected in Webtrends Analytics Accounts.

To view your server calls balance:

In the left pane of Webtrends Accounts, select **Account Usage > Summary**. Use this dialog to view:

- Server calls collected for the current month to date
- Server calls collected for the calendar year to date
- Server calls collected during the life of your account

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section

Chapter 2

Client-Side JavaScript Integration

Webtrends On Demand and SmartSource Data Collector (SDC) use a special JavaScript tag that you place on your web pages to collect activity. When a visitor accesses a page from your web site, this JavaScript tag initiates interactions between the visitor's browser and either Webtrends On Demand (for service only) or SDC (for software). The tag collects data about the visitor's browser and activity and transmits this information to Webtrends On Demand or SDC. For Webtrends Analytics reports, an analysis engine aggregates this data with that of all other visitors to your site, stores it, and makes it available to you in reports that you can view and download. For Webtrends Marketing Warehouse, an analysis engine analyzes this data, which is stored in the Marketing Warehouse and made available to you in Webtrends Explore and Webtrends Visitor Intelligence.

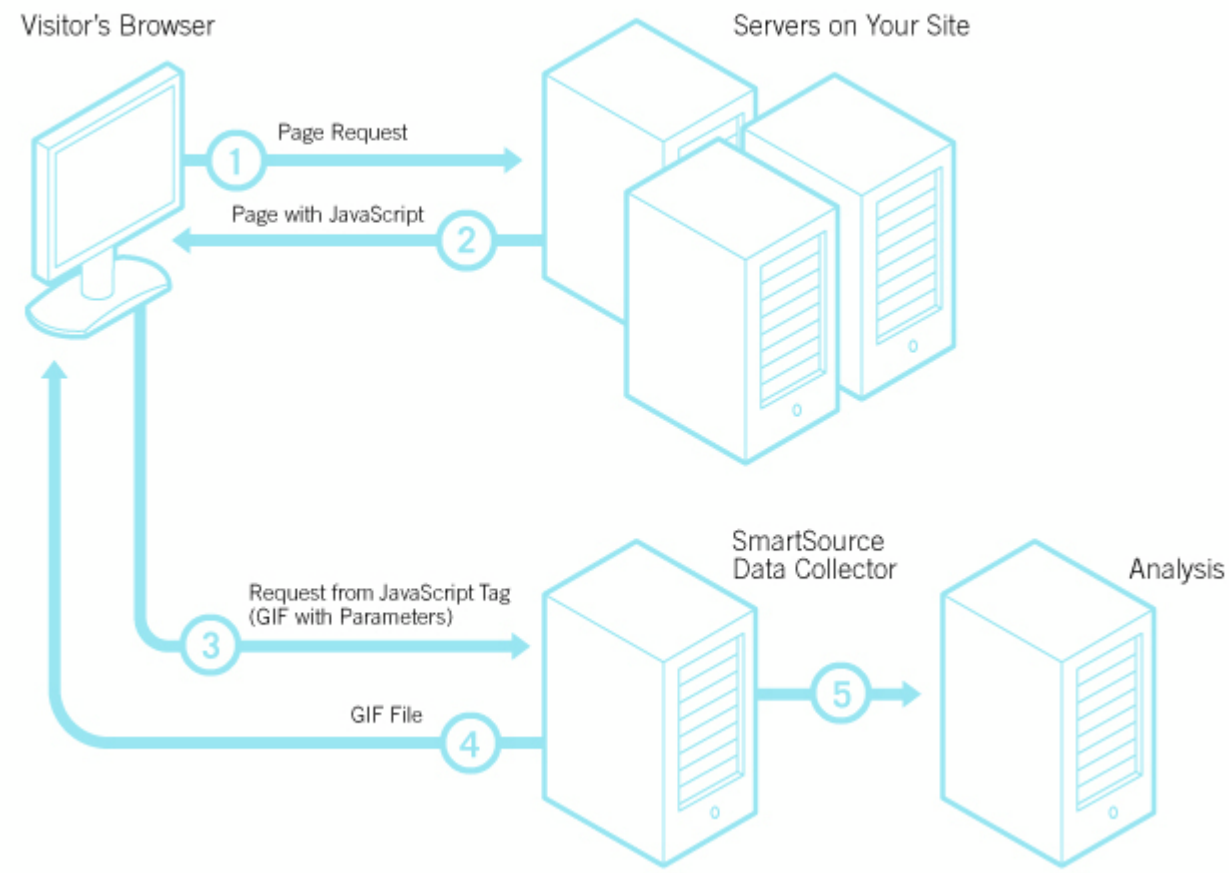
This chapter explains how client-side data collection works, and describes how to customize your Webtrends JavaScript tag using META tags to collect the data that interests you.

Interactions Between the Client Browser and SDC

After deploying the Webtrends JavaScript tag on your web pages, you are set up for a series of interactions between your visitors, your web site, and SmartSource Data Collector. The first interaction is between your visitors and your web site.

When a visitor accesses a page from your web site, the JavaScript tag initiates an interaction between the visitor's browser and sends that data to the SmartSource Data Collector (SDC), which Webtrends On Demand uses to collect web activity data. The data is analyzed, stored, and made available to you in your Webtrends Analytics reports, Webtrends Explore, or Webtrends Visitor Intelligence.

The following illustration shows an overview of the interaction process.



The following interactions take place:

1. A visitor wants to view a page on your site. This initiates a page request to your web server.
2. Your web server sends the page to the visitor that contains your Webtrends JavaScript tag.
3. The JavaScript tag triggers a request for a GIF file with additional tagging parameters and cookie attached. This image request is sent to the SmartSource Data Collector or Webtrends On Demand.
4. The GIF file and cookie is sent to the visitor.
5. The image request with the parameters is collected and analyzed.

All interactions between your visitors and Webtrends take place at the browser on the client side (your visitor's side). With client-side integration, there is no interaction between Webtrends On Demand and your web servers.

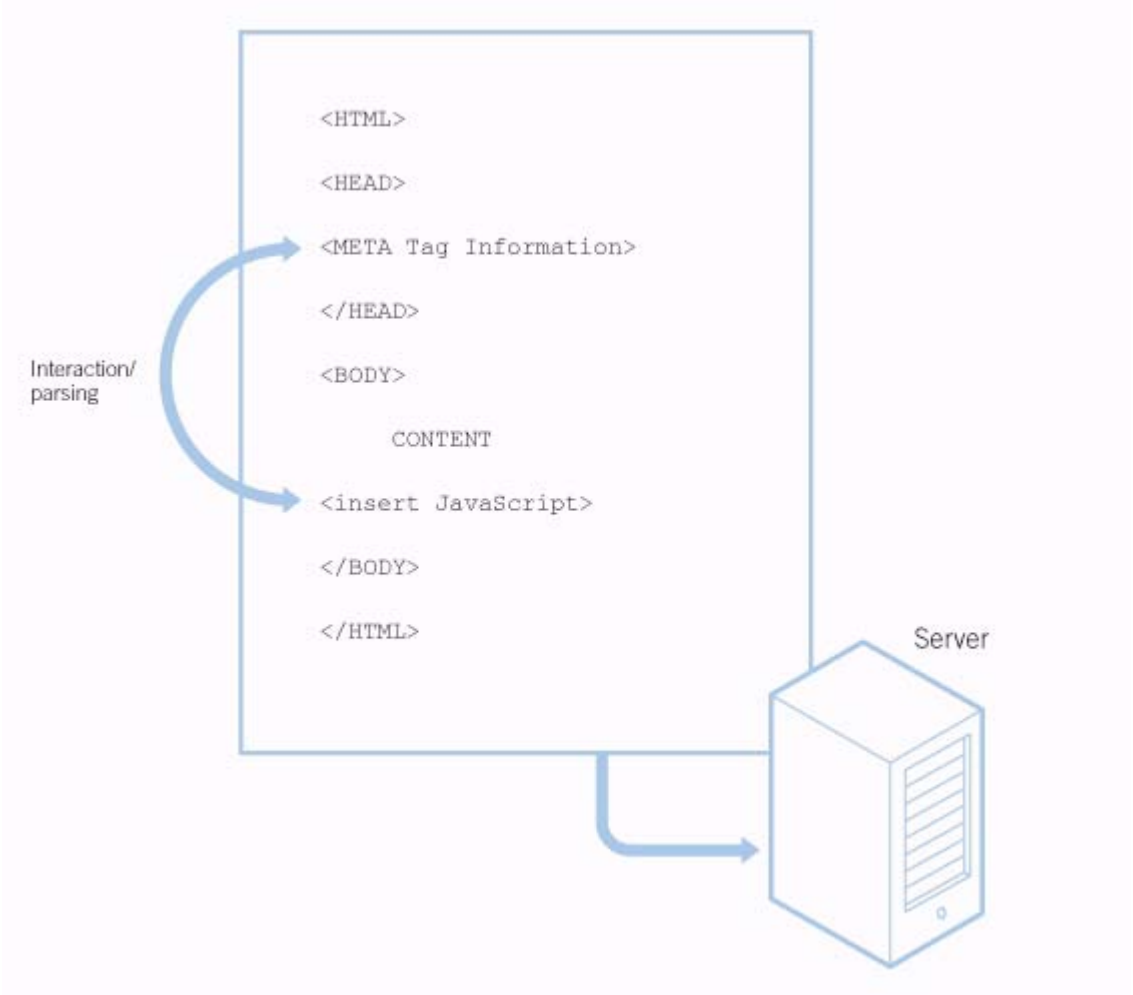
Customizing Your META Tags

After you put the Webtrends JavaScript tag on your web page, you may also modify the META tags on your web pages because the tag interacts with the META tags and stores the resulting information in SmartSource Data Collector log files.

The following sections discuss all the META tags that interact with the JavaScript tag. Note that the META tags do not have to appear in a particular order in the web page. However, they are presented here in an order that helps you more readily understand the structure of your web page.

Webtrends offers one of a few data tagging implementations that segregate page-specific information from the main script, maximizing code modularity and reuse.

The following illustration provides an overview of the JavaScript tag and META tag interaction.



General META Tag Information

The general syntax of the supported META tag is as follows:

```
<META NAME="name" CONTENT="content">
```

Include the META tag between the <HEAD> and </HEAD> tags.

The *name* represents the parameter name.

The *content* represents the parameter value.

Many META tags support more than one parameter. Separate multiple parameters by a semicolon ";" as shown in the following example:

```
<META NAME="name" CONTENT="content1;content2;content3...">
```

META Tag Descriptions

The following subsections discuss the META tags that may be needed to add or modify on your web pages.

Note that all of the META tags that you use begin with WT., for example, WT.cg_n, which you can use to track content groups.

Tracking Content Groups

Webtrends can report on visitors according to the content group and content subgroup of the pages they visit. To do so, your site must capture the content group and subgroup in a parameter when the page is visited, and then pass the parameter values to Webtrends. Content subgroups are optional.

The following META tags track content groups:

<META NAME="WT.cg_n" CONTENT="Name">
Defines the name of the content group.

<META NAME="WT.cg_s" CONTENT="subName">
Defines the name of the content subgroup. This tag is optional.

Sample: Single Content Group and Subgroup

A university web site reports the number of visitors to its pages. The web site denotes the content group DegreeReq for each page that describes the requirements. The web site then assigns a subgroup designation for each page about a particular field of study, for example, Lit for Literature.

Using the WT.cg_n and WT.cg_s tags, your META tag would look like this:

```
<META NAME="WT.cg_n" CONTENT="DegreeReq">
<META NAME="WT.cg_s" CONTENT="Lit">
```

Example: Multiple Content Groups

Building on the single content group example, the Admissions Department is also interested in visitors to these pages. An additional group can be added so that the pages are reported for both content groups. Note that Math has been added as a subgroup. Multiple content groups and subgroups are separated by semi-colons.

```
<META NAME="WT.cg_n" CONTENT="DegreeReq;Admissions">
<META NAME="WT.cg_s" CONTENT="Lit;Math">
```

Sample Log File

After the JavaScript interacts with this META tag information, it generates logs that look something like this:

```
2008-10-24 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//
www.biguniversity.edu&WT.cg_n=DegreeReq;Admissions&WT.cg_s=Lit;Math....
```

The italicized text represents the META-related order information captured by the JavaScript and placed in the log file.

Tracking Servers

If your site is hosted on multiple servers, a server cluster, or a server farm, and you want to evaluate the performance of your load balancer, Webtrends can track page views for each server. To do so, populate the following META tag on all pages on each server:

<META NAME="WT.sv" CONTENT="name">

Defines the name of the machine that serves the web page.

If you have two servers (Server1 and Server2), you would make two copies of the META tag and designate `CONTENT="Server1"` for deployment to pages on the first server and `CONTENT="Server2"` for deployment to the same pages on the second server.

For a server farm, you can extract the value of the built-in server name and dynamically assign it to the META tag using server-side scripting.

Sample Log File

After the JavaScript interacts with this META tag information, it generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phonedealer.com&WT.sv=Server1 ...
```

The italicized text represents the META-related order information captured by the JavaScript and placed in the log file.

Tracking Marketing Campaigns

Webtrends can report visitor activity that relates to a marketing campaign. You can place the following META tags on the landing page to identify the name of the campaign and the type of campaign.

<META NAME="WT.mc_id" CONTENT="Campaign ID">

Identifies the ID of the marketing campaign.

Landing Page

The landing page is the first page that visitors see when they visit your site. Normally, this is your home page, but for effective marketing campaign tracking, you can bring visitors to a page exclusively used for your marketing campaign.

Example Marketing Campaign

To attract new students, a university launches a marketing campaign by sending recruitment email to all graduating high school seniors in a metropolitan area. The email links to a special landing page in the university's web site, containing the following META tag to track marketing campaigns.

```
<META NAME="WT.mc_id" CONTENT="1X2GG34">
```

The Campaign ID 1X2GG34 represents recruits to be contacted by email.

Sample Log File

After the JavaScript interacts with this META tag information, SDC generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.biguniversity.edu&WT.mc_id=1X2GG34 ....
```

The italicized text represents the META-related order information captured by the JavaScript and placed in the log file.

Tracking Profile/Subprofile Generation

If you are using parent/child profiles, the split profile META tag allows you to identify the pages that are associated with each child profile. In addition to putting this META tag on your web pages, you also create a parent profile in the UI that specifies the WT.sp parameter as well as the values that identify your child profiles. The child profile values that you specify should match the values of the `CONTENT=` string. Use the following META tag to track child profiles.

`<META NAME="WT.sp" CONTENT="profile name">`

Defines the identification string for creating the child profile, and is used to track child profile activity. For more information about parent/child profiles, see the Administration Help. For more information about parent/child profiles, see the Administration Help.

Example Tracking Profile/Subprofile Generation

Suppose you are an Internet Service Provider and you collect web traffic data for three different customers whose account numbers are: 11111, 22222, 33333. You can use Parent/Child profiles to split-out traffic based on a Webtrends Query Parameter `WT.sp`.

```
<META NAME="WT.sp" CONTENT="11111">
```

The child profile name 11111 will be created automatically by the Webtrends Analytics. Traffic from pages containing this META tag will be directed into this profile.

Sample Log File

After the JavaScript interacts with this META tag information, it generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phonedealer.com&WT.sp=Wireless%20Phones ...
```

The italicized text represents the META-related order information captured by the JavaScript and placed in the log file.

Tracking Revenue

To track Commerce revenue, configure your web site to populate the META tags on your confirmation pages. Your web site captures transaction information, typically using an order form. Configure your site to pass the values from the form to a META tag so that Webtrends can track the transactions, aggregate them, and include them in your reports.

To track revenue, include the following META tags:

`<META NAME="WT.pn_sku" CONTENT="ProductSKU">`
Identifies the SKU of the product. Use semicolons to pass multiple SKUs for the order.

Note
`WT.pn` has been replaced with `WT.pn_sku`. `WT.pn` can still be used, but it does not work with product SKUs.

`<META NAME="WT.pc" CONTENT="ProductCategory">`
Defines the category of the product. Use semicolons to pass multiple categories.

`<META NAME="WT.tx_u" CONTENT="units">`
Defines the quantity purchased. If the order contains multiple products, pass a semicolon-delimited list of units.

`<META NAME="WT.tx_s" CONTENT="subtotal">`
Defines the total cost for each `WT.pn_sku` value passed. If the order contains multiple SKUs, pass a semicolon-delimited list of values for this parameter. However, do not pass a dollar sign ($) or comma(,) in the subtotal variable.

Example of Multiple Usage

You can pass multiple orders to the variables in the META tags by using a semicolon-delimited list. For example, the following tag represents two products:

```
<META NAME="WT.pn_sku" CONTENT="1X11GG34;2YR5R53">

<META NAME="WT.tx_u" CONTENT="2;7">

<META NAME="WT.tx_s" CONTENT="130.00;150.00">
```

Sample Log File

After the JavaScript tag interacts with these META tags, it generates log files that may look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phonedealer.com&WT.pn_sku=1X11GG34;2YR5R53
&WT.tx_u=2;7&WT.tx_s=130.00;150.00 ....
```

The italicized text represents the META-related order information captured by the tag and placed in the log file.

Note
In the log file `%20` is an ASCII representation for a blank space.

Tracking Shopping Cart Activity

You can use META tags that track shopping cart activity. At analysis, these tags are interpreted as steps along the path that leads to a successful completion of the shopping activity. Webtrends preconfigured Purchase Conversion Funnel includes four steps. For more information, see .

Tracking On-Site Advertising

Visitors often view advertisements that they do not necessarily click on. You can use On-Site Advertising to determine the number of visitors to your web site who view particular ads. With this feature you can produce advertising reports for each of your clients.

If you are selling advertising space on your web site, for example, you can collect traffic statistics to help determine pricing schedules.

The following META tag tracks advertising views:

`<META NAME="WT.ad" CONTENT="name">`
This metatag defines the name of the advertisement viewed on this page. You can designate multiple ad views using semicolons.

Example On-Site Advertising

To attract business to your online hotel reservation system, you place a promotional advertisement on a page. You want to find out how many visitors viewed this advertisement. You can use the Webtrends Query Parameter WT.ad for this purpose.

```
<META NAME="WT.ad" CONTENT="Weekend Special Rate">
```

The advertising name “Weekend Special Rate” represents an advertisement that was viewed on the page.

Sample Log File

After the JavaScript tag interacts with this META tag information, SDC generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phonedealer.com&WT.ad=Weekend%20Special%20Rate ...
```

The italicized text represents the META-related order information captured by the JavaScript tag and placed in the log file.

Tracking Advertising Clicks

When a visitor to your site clicks on an ad, that action is referred to as an *Ad Click*. The following META tag tracks advertising clicks:

<META NAME="WT.ac" CONTENT="name">

Defines the name of the advertisement clicked to reach a particular web page. The Ad Click must contain an external redirect back to the client. The redirect needs to include the necessary code to generate a hit to the SDC server. You can designate multiple Advertising Clicks using semicolons.

Example Tracking Advertising Clicks

You have a banner ad that links to another site with weekend getaway ideas and you receive payment for each referral to the site. You can use the Webtrends Query Parameter `WT.ac` to track clicks on the banner ad.

```
<META NAME="WT.ac" CONTENT="Weekend Pointer">
```

The advertising name “Weekend Pointer” represents an advertisement that was clicked on your page.

Sample Log File

After the JavaScript tag interacts with this META tag information, SDC generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phone dealer.com&WT.ac=Weekend%20Pointer ...
```

The italicized text represents the META-related order information captured by the JavaScript tag and placed in the log file.

Tracking Customized URLs

Webtrends reports only the base URL when it compiles reports on pages identified by URLs. “Top Pages” and “Page Views Trend” are reports that use page URLs. These reports could become overwhelming and meaningless because URL parameters are used to carry many different kinds of information in addition to dynamic page identification. There can be many variations to a single page URL. The parameters make it seem as if there are many unique URLs when there is only one, the base URL.

Dynamic pages are an exception. With dynamic pages, Webtrends counts each URL with different parameters. By customizing a URL, you can track dynamic pages by changing the URL before it is passed to Webtrends.

META Tags That Customize URLs

The following META tags allow you to customize URLs:

<META NAME="DCS.dcsuri" CONTENT="*uri-stem*">

Assigns the information to the cs-uri-stem field of the log file.

Sample Log File

If the URL looks like this without any customization:

```
http://www.asite.com/browse.asp?UID=14&Cat=Rock&Artist=Your_Band&Album=Jar_of_Flies
```

the Top Pages report only shows `http://www.asite.com/browse.asp`.

If you write a server-side script to dynamically convert the parameters to page names, the URL might look like this:

```
http://www.asite.com/Rock/Your_Band/Jar_of_Files.asp.
```

Place that URL in a META tag as follows:

```
<META NAME="DCS.dcsuri" CONTENT="http://www.asite.com/Rock/Your_Band/
Jar_of_Files.asp">
```

After the JavaScript interacts with this META tag information, it generates log files that look something like this:

```
2007-08-10 00:06:06 192.168.100.40 - web1 GET /Rock/Your_Band/Jar_Of_Flies.asp...
```

The italicized text represents the META-related URL information captured by the JavaScript and saved in the log file.

Tracking Page Titles

You may want to modify a page title before sending it to Webtrends in the following cases:

- You are dealing with dynamic content pages identified by URL parameters, and the page title represents the title of the base URL page rather than the dynamic content page.

Unless you modify the page titles, all pages have the same title in the reports.

- All pages have been assigned the same title, for reasons of style or company policy.

Even though URLs are displayed in addition to page title, the entire URL cannot be depended upon to distinguish one page from another.

Use server-side scripts to change the title to something that reflects the content of the pages so that you can identify them in reports. Next, pass the customized page titles to Webtrends, using the following META tag:

`<META NAME="WT.ti" CONTENT="title">`
Defines the name of the title for this page.

Example Page Title

You would like to specify the name of the page that is displayed on your reports. You are using a hosting service that does not allow you access to the <TITLE> tag on your page. You can assign a title for the page using the Webtrends Query Parameter `WT.ti`.

```
<META NAME="WT.ti" CONTENT="Advertising">
```

The page title “Advertising” represents the name of the page that is displayed on your reports.

Sample Log File

After the JavaScript tag interacts with this META tag information, SDC generates log files that look something like this:

```
2007-03-04 00:08:18 proxy7.hotmail.com - W3SVC3 web1 192.168.1.1 GET /ads/default.asp
redir=products&ad=http%3A//www.phonedealer.com&WT.ti=Advertising ...
```

The italicized text represents the META-related order information captured by the JavaScript tag and placed in the log file.

Inserting the Tags

You can insert the JavaScript tag in several ways: by copying appropriate tag versions to each of your web site pages, using server-side includes on appropriate web servers, or by inserting the tag in the footer template. The method depends on your needs, your maintenance practices, and the programming resources available to you.

Once you have deployed the tag to your site, you can begin viewing reports of your visitor activity and of the revenue your site generates. Reports are typically available 24 hours after the tags are deployed.

Copying the Tag to Each Page

Copy the same tag or individually modified copies of the code to your web site pages. To minimize the impact on your web site, place the tag as close as possible to the `</BODY>` tag.

Using Server-Side Include Files

Server-side includes (SSI) are enabled by default on Internet Information Server and Apache web servers. You can either configure the server to run SSI on all files with the extensions you use for your web pages (`.htm`, `.html`), or you may need to change your page extensions (to `.stm`, `.shtm`, and `.shtml`, for example).

To set up the include file and the include statements:

1. Place the include file containing the JavaScript tag in it in a location accessible to every page of your site.
2. Place an include statement on all of your web site pages. Be sure to use the correct file extensions. For example, if your include file is named `code_include.inc` and located in the `mysite` directory, you place the following include statement on your web pages:

   ```
   <!--#include virtual="/mysite/code_include.inc"-->
   ```

Using Footer Templates

If your web site uses header and footer templates, you can place the JavaScript tag in the footer template. Place it as close as possible to the `</BODY>` tag.

Keep in mind that a commerce confirmation page must include the revenue tracking code. You need to configure the confirmation page to include both the JavaScript tag and the revenue tracking code. Keep in mind that other pages cannot include the revenue tracking code.

Tagging Best Practices

Web Page Editor Issues

A number of HTML editors actually modify your HTML code and can break JavaScript. Make sure that your HTML editor does not modify the tag in any way.

Specify Character Sets On Tagged Pages

As a best practice, you should include a character set META tag on the pages that have your Webtrends JavaScript tag. For example, `<META HTTP-EQUIV="Content-Type" CONTENT="text/html; charset=iso-8859-1">`.

If you don't specify the character set, a browser may use the character set defined on the previously viewed web site. If that character set does not match the one you intend to be used on your own, your tags may not be consistently encoded. The resulting reports would show a single web page as several different web pages, and the text for the pages that were not encoded correctly may not be properly displayed.

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Chapter 3

Server-Side Integration for Webtrends Analytics On Demand

Webtrends Analytics On Demand is typically implemented by placing a special piece of JavaScript code on each Web page that you want to track, as described in . As the Web page is loaded in a visitor's browser, the Webtrends JavaScript tag gathers data about the visitor's browser and activity, and transmits this information to the Webtrends SmartSource Data Collector (SDC) servers. This data is transmitted through an HTML information request (`statse.webtrendslive.com/ .../dcs.gif`) with the visitor information added to the query string.

The JavaScript tag usually handles the task of automatically gathering the information, formatting the HTTP request, and then passing the data to the SmartSource Data Collection servers. But in some cases, the use of the JavaScript tag may not be the best method for implementing Webtrends On Demand tracking. The following are some examples:

- Your application server and/or content management server does not support the inclusion of JavaScript.
- You are tracking commerce or other dynamic events, where the information you need to track is known to the application server, but not to the HTML page.
- Your visitors' browsers do not support JavaScript, or the visitors have disabled JavaScript support in their browsers.

In the situations outlined above, it may be necessary to use server-side code to compose the HTTP request.

Sample HTTP Request

The domain `statse.webtrendslive.com`, your Site ID, and the page `/dcs.gif` remain the same on each of your pages. You can customize the parameters for the request according to the information that you want to track with your server-side code. For more information about customizing parameters for your implementation, see "Customizing the Request" on page 19.

The following URL is an example of the typical information that should be supplied as part of the HTTP request used to collect your site's data.

Example 1. HTTP Request

```
http://statse.webtrendslive.com/@MySiteID@/dcs.gif?dcsuri=
MyPage.asp&dcsdat=1068089121752&WT.ti=My%20Page%20Title
```

The following table explains each of the components in this request.

Code Sample	Description	Sample Value
`http://statse.webtrendslive.com/`	The Internet address of the Webtrends SmartSource data collection servers.	`http://statse.webtrendslive.com/`
`@MySiteID@`	The SiteID given to you by your Webtrends Integration team	`@MySiteID@`
`/dcs.gif`	The image that is used in transmitting your site's data	`/dcs.gif`
`dcsuri`	The page that is being browsed	`MyPage.asp`
`dcsdat`	A random number that prevents the caching of the image	`1068089121752`
`WT.ti`	The page title that appears in your reports.	`My%20Page%20Title`

The request is implemented as part of an `IMG` tag. To track browsers that do not support JavaScript, insert the composed request into the `<NOSCRIPT>` section of your pages.

Example 2. NOSCRIPT Code

```
<NOSCRIPT>

<IMG BORDER="0" NAME="DCSIMG" WIDTH="1" HEIGHT="1"

SRC="http://statse.webtrendslive.com/@MySiteID@/
dcs.gif?dcsuri=My%20page.asp&dcsdat=1068089121752&WT.ti=My%20Page%20Title">

</NOSCRIPT>
```

To track commerce transactions, include the request in the <BODY> area of the page as shown in the following example.

Example 3. BODY Code

```
<HTML>
<HEAD>
<BODY>
```

```
<IMG BORDER="0" NAME="DCSIMG" WIDTH="1" HEIGHT="1"
SRC="http://statse.webtrendslive.com/@MySiteID@/
dcs.gif?dcsuri=My%20page.asp&dcsdat=1068089121752&WT.ti=My%20Page%20Title&CommerceValu
es=MyCommerceValues">

</BODY>
</HTML>
```

Customizing the Request

This section discusses the types of information that are typically passed as part of the tag request, and important considerations when developing the strategy for your server-side implementation. Although server-side implementations vary depending on the application server or content delivery server, this section provides example code.

Protocol

When using the standard Webtrends On Demand JavaScript tag, the tag automatically detects whether the page in which it is included is secure or non-secure. Your server-side implementation should also ensure that the protocol of the composed request is the same as the page in which it is included. This task is handled by the following code in the standard JavaScript tag.

```
"http"+(window.location.protocol.indexOf('https:')==0?'s':'')+":/"
```

URLs

The URL is a required parameter that is passed using the `dcsuri` variable. This value can be read from the server using `Request.ServerVariables("URL")` in ASP, or `request.getRequestURI()` in Java.

Note

In Java, you also need to enumerate all of the parameters using `request.getParameternames()`, and include those in your `dcsqry` value. Example 5 demonstrates this code.

Page Titles

The value of the page name that is displayed in your reports is passed using the `WT.ti` query parameter. `WT.ti` is not a required parameter.

Referrers

The Referrer value is passed using the `dcsref` variable. This value can be read from the server using `Request.ServerVariables("HTTP_REFERER")` in ASP, or `request.getHeader("Referer")` in Java. Referrers are important in understanding which site visitors were browsing prior to clicking on a link and being directed to your site, such as search engines, search phrases, or affiliates.

Escaping Characters

Please note that in the request used in Example 1, the spaces are *escaped* or *URL encoded*. The escaped value for the space is `%20`. Using escaped characters allows you to pass special characters such as spaces or slashes properly to the SmartSource Data Collection servers. All special characters passed in the parameter values must be escaped.

Avoiding Cached Requests

Most web browsers and servers keep cached, or locally saved copies of content that visitors have recently viewed to display content more quickly when visitors return to the page. If the image that used in the server-side request is cached, your web site activity tracking will be inaccurate. To ensure the most accurate data collection, take the necessary steps to prevent the caching of the image that is used by the request. In the standard JavaScript tag, you accomplish this by passing a serialized version of the current time to the `dcsdat` variable, as shown in Example 4. Your server-side code should also pass a random number using this variable to avoid caching issue.

Example 4. Populating DCSDAT

The JavaScript tag includes the following code to get the current time.

```
DCS.dcsdat=Current.getTime();
```

Additional Query Parameters

The following table lists optional parameters that may be included in your server-side requests to automatically configure deeper levels of insight in your reports.

Parameters	Type	Usage
`WT.sv`	Server	The Web server that delivers the content
`WT.cg_n`	Content	Content group to which the page belongs
`WT.cg_s`	Content	Sub-content group to which the page belongs
`WT.mc_id`	Marketing	ID of the marketing campaign
`WT.ad`	Marketing	Ad displayed on the page
`WT.ac`	Marketing	Ad that was clicked
`WT.si_n`	Conversion	Name of the scenario being measured
`WT.si_x`	Conversion	Step number of the scenario being measured

Parameters	Type	Usage
`WT.si_p`	Conversion	Name of the step in the scenario being measured
`WT.pn_sku`	Commerce	Product SKU
`WT.tx_s`	Commerce	Sub-total of items being purchased
`WT.tx_u`	Commerce	Quantity of items being purchased

For more information about the query parameters recognized by Webtrends On Demand, see . Another useful chapter is which covers how to implement the query parameters through META tags in HTML pages. You can also append custom parameter name-value pairs that are specific to your web site to the composed image request for data collection. You can then include these custom values in reports by creating a custom reports.

Sample Java Code

Server-side implementations vary greatly depending on the server technology, but by constructing the server-side requests as described in the previous section, you can implement tracking of browsers or sites that do not support JavaScript.

The following example shows how server-side code was used to instrument an IBM WebSphere site. In addition to the standard page information, the code also retrieves the UserID and Company from the WebSphere database and passes that data as custom parameters.

Example 5. JAVA Code

```
<%
//
// Make an image request of SDC Server with appropriate query parms
// Uses specific classes in engine to grab company/sessionID type parms
//

// Grab URI StringBuffer
p = new StringBuffer("dcsuri="+request.getRequestURI());

// Grab the Query string / form parameters
StringBuffer queryString = new StringBuffer();
java.util.Enumeration e = request.getParameterNames();
boolean started = false;
while(e.hasMoreElements())
{
String name = (String) e.nextElement();
if (!started)
{
started = true;
queryString.append(name+"%3D"+request.getParameter(name));
}
else if (null != request.getParameter(name))
{
queryString.append("%38"+name+"%3D"+request.getParameter(name));
}
}
if (started)
```

```
p.append("&dcsqry="+queryString.toString());

// Grab the User id and company (if available)
oem.edge.entitlement.engine.EntitlementEngine ee =
(oem.edge.entitlement.engine.EntitlementEngine)request.getAttribute("entEngine");
if (null != ee)
{
p.append("&userId="+ee.getUserId());
oem.edge.entitlement.engine.UserInfo userInfo = ee.getUserInfo();
if (null != userInfo)
p.append("&company="+userInfo.getUsrCompany());
}
// Session id (if available)
if (null != request.getSession(false))
{
HttpSession reqSession = request.getSession(false);
p.append("&sessionId="+reqSession.getId());
}
// Locale (language)
p.append("&lang="+request.getLocale().toString());

// Date
p.append("&dcsdat="+ (new java.util.Date()));
%>

//Send the request to the SDC servers
<img border="0" name="DCSIMG" width="1" height="1" src="http://
statse.webtrendslive.com/@MySiteID@/dcs.gif?<%= p %>"/>
```

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7	March, 2009	Added Document Revision History section.

Chapter 4

Customizing JavaScript Tags

The Webtrends Tag Builder produces code known as the JavaScript tag. The Webtrends JavaScript tag contains code that enables you to collect visitor data after the tag is implemented on your Web site. This chapter discusses the contents of the tag before you modify it, and covers more advanced ways you can modify it to get additional information.

If you use Webtrends On Demand or SmartSource Data Collector and want to learn to tag your Web site and create META tags that automatically collect visitor information, see .

Understanding the Basic JavaScript Tag

This section breaks the tag into functional parts and describes how the tag performs data collection.

The JavaScript tag is divided into the following two files:

- **`webtrends.html`**, which contains the markup you place in the HTML body of your Web pages.
- **`webtrends.js`**, which contains the JavaScript file that you host on your Web server.

Inline HTML (webtrends.html)

The markup that goes in the HTML body of your Web pages is enclosed inside HTML comments similar to the following:

```
<!-- START OF SmartSource Data Collector TAG -->
<!-- Copyright (c) 1996-2008 WebtrendsWebtrends Inc.  All rights reserved. -->
<!-- Version: 8.5.0 -->
<!-- Tag Builder Version: 2.1.0  -->
<!-- Created: 9/23/2008 10:32:55 AM -->

…
<!-- END OF SmartSource Data Collector TAG -->
```

The markup is comprised of three HTML <*script*> blocks, and one HTML <*noscript*> block.

External Script File

The first HTML <*script*> block contains a **`src`** attribute referencing an external script file named **webtrends.js**:

```
<script src="webtrends.js" type="text/javascript"></script>
```

Note that the file location specified in the `src` attribute can be a full or relative URI. This depends on where it resides on your Web server.

The **`webtrends.js`** file contains the object constructor function named Webtrends. This function is used to store, manipulate, and send data to Webtrends data collection servers.

Embedded JavaScript

The second and third HTML <*script*> blocks contain embedded JavaScript. These blocks must remain separate for the tag to function properly (that is, they cannot be combined into one block).

CDATA

The embedded JavaScript in both blocks is wrapped inside CDATA sections:

```
//<![CDATA[
…
//]]>>
```

CDATA sections are used to make the tag XHTML compliant. Certain characters found in embedded JavaScript will cause XML parsers to fail (">" and ";"). XML parsers ignore everything inside a CDATA section, so the tag will not cause a parsing failure.

The CDATA sections are commented out using JavaScript single-line comments (`"//"`). This prevents the JavaScript interpreter from throwing an error.

If XHTML compliance is not a requirement, then the CDATA sections may be removed.

Initialization

The second HTML <*script*> block performs tag initialization.

```
var _tag=new Webtrends();
_tag.dcsGetId();
```

The first line instantiates an object named `_tag` using the Webtrends object constructor function (defined in **webtrends.js**). The `_tag` object serves as a namespace that we can use to access properties and invoke methods.

The second line invokes a method of the `_tag` object named `dcsGetId()`. This method performs an inline JavaScript request for the file `wtid.js`. This request is used to obtain a unique visitor identifier that is persisted in the Webtrends first party cookie.

Collection

The third HTML <*script*> block performs data collection.

```
_tag.dcsCollect();
```

This line invokes a method of the `_tag` object named `dcsCollect()`. This method performs an image request for the file `dcs.gif`. The image request is the mechanism used to pass information to Webtrends data collection servers.

Script Disabled

The <*noscript*> block contains an inline image request similar to the following:

```
<div><img alt="DCSIMG" id="DCSIMG" width="1" height="1" src="http://statse.webtrend-
slive.com/dcs5w0txb10000wocrvqy1nqm_6n1p/njs.gif?dcsuri=/
nojavascript&amp;WT.js=No&amp;WT.tv=8.5.0"/></div>
```

The request is hard-coded to contain a special URI-stem and a minimal subset of Webtrends Query Parameters. You may add parameters if necessary.

The `dcsuri` parameter is logged as the URI-stem with the value of `/nojavascript`.

Webtrends Analytics can be configured to report on this filename. `WT.js=No` tells Webtrends Analytics that the page view was from a non-JavaScript enabled browser.

The <*img*> tag is wrapped inside a <*div*> tag to make the tag XHTML compliant. If XHTML compliance is not important to you, then the <*div*> tag may be removed.

External JavaScript File (webtrends.js)

The **webtrends.js** file contains an object constructor function named Webtrends. This function can be thought of as a class containing properties and methods. All properties and methods are publicly available, and an object of this class can be instantiated using the new operator. Once you have instantiated an object, you can modify properties, and invoke methods.

Construction

The inline HTML portion of the tag uses the JavaScript new operator to call the object constructor function as follows:

```
var _tag=new Webtrends();
```

In this case, the resultant object is named `_tag`. All properties and methods are accessed using the `_tag`. qualifier.

During construction, properties are initialized using settings specified during tag creation process. For example, suppose you specified a `dcsID` value of dcs5w0txb10000wocrvqy1nqm_6n1p when you created the tag. At construction time, the `dcsID` property is initialized as follows:

```
this.dcsid="dcs5w0txb10000wocrvqy1nqm_6n1p";
```

This property can be accessed using the `_tag` qualifier as follows:

```
alert("The DCSID is:" + _tag.dcsid);
```

Properties

The Webtrends function contains several properties. All properties are public in scope. Some properties are designed to be modified, others are read-only. Some properties are only present when applicable options were selected during tag creation. As the tag evolves over time, new properties are likely to be added.

To access a property from within a Webtrends constructor function method, prefix the property with `this.`. For example, the `dcsTag` method accesses the `WT.ad` property as follows:

```
this.WT.ad="";
```

To access a property globally, prefix the call with `_tag.`. For example, a custom parameter could be set from the <*script*> block in the HTML inline as follows:

```
_tag.DCSext.team="Mariners";
```

User Modifiable Properties

Property	Type	Present when…	Description
dcsid	String	always	SmartSource site id (DCSID)
domain	String	always	Data collection server domain name or IP address
timezone	Number	always	Timezone of customer web site (integers between -12 and 12)
fpcdom	String	always	Webtrends First Party Cookie domain attribute
onsitedoms	String	track clicks to download, offsite, or anchor links are enabled	Onsite domain names (comma separated list or regular expression)
downloadtypes	String	track clicks to download links is enabled	Download file types (comma separated list)
rightclicktypes	String	track right clicks to download links is enabled	Right click file types (comma separated list)
navigationtag	String	any click event tracking enabled and navigation area selected is DIV or TABLE	Navigation area that encloses a link or button (None, DIV, TABLE)
customerfpc	String	use customer provided cookie is enabled	Name of cookie used to seed visitor id value (WT.co_f)
adclickparam	String	create ad view from links containing this parameter is enabled	Name of parameter used to designate an ad click (default is WT.ac)

metanames	String	capture these custom meta tags is enabled	Names of custom meta tags (comma separated list)
trackevents	Boolean	any click event tracking enabled	Programmatically enable/disable click event tracking on a page
evi	Object	set query parameter using this cookie is enabled	Object that contains properties for cookie name (cookie), query parameter to set (qp), cookie crumb (crumb), cookie crumb separator (sep)
enabled	Boolean	always	Programmatically enable/disable data collection by tag
i18n	Boolean	always	Programmatically enable/disable extended internationalization support
fpc	String	always	Webtrends First Party Cookie name

Read Only Properties

Property	Type	Present when…	Description
that	Object	always	Private variable used to store “this” object for closure
DCS	Object	always	Properties mapped to SDC-specific query parameters
WT	Object	always	Properties mapped to Webtrends Query Parameters
DCSext	Object	always	Properties mapped to custom query parameters
images	Array	always	Image array that holds 1x1 pixel request (dcs.gif)
index	Number	always	Index into image array
qp	Array	assign custom query parameters to Webtrends parameters enabled	Temporary storage of custom query parameters assigned using the dcsQP() function
exre	Object	always	Anonymous function containing regular expression
re	Object	always	Anonymous function containing regular expression list used to URL encode query parameter values
vtid	String	use alternate Webtrends Visitor Id is enabled	Alternate visitor id for Warehouse seeded by query parameter or cookie

Methods

The Webtrends constructor function contains several methods. All methods are public in scope. Some methods are only present when applicable options were selected during tag creation. As the tag evolves over time, new methods are likely to be added.

Generally, the method names are prefixed with `dcs`. To access a method from within another method, prefix the call with `this.`. For example, the `dcsTag` method calls the `dcsCreateImage` method as follows:

```
this.dcsCreateImage(P);
```

To access a method globally, prefix the call with _tag.. For example, the HTML inline <script> tag calls the dcsCollect method as follows:

```
_tag.dcsCollect();
```

The following table contains the high level methods of consequence contained in the tag.

Method	Return type	Parameter list	Description
dcsGetId	none	none	Performs an inline JavaScript request to data collection servers. If no Webtrends First Party Cookie is present on the client, a request is for the file wtid.js occurs. This file contains an unique visitor identifier that is persisted in the Webtrends First Party Cookie and passed via a Webtrends Query Parameter (WT.co_f).
dcsGetCookie	String containing cookie value or null	name - String containing cookie name	Reads a cookie name and returns its value.
dcsGetCrumb	String containing cookie crumb value	cval - String containing cookie value crumb - String containing cookie crumb name sep - String containing cookie crumb separator	Reads a cookie crumb from Webtrends First Party cookie and returns the crumb value.
dcsGetICrumb	String containing cookie crumb value	cval - String containing cookie value crumb - String containing cookie crumb name	Reads the “id” cookie crumb from Webtrends First Party cookie and returns the crumb value.

dcsIsFpcSet	Number indicating if cookie was set (0=success, non-0=failure)	name – String containing cookie name id – String containing id crumb value lv – String that is the lv crumb value ss – String containing ss crumb value	Tests to see that the Webtrends First Party Cookie was successfully created.
dcsFpc	none	none	Sets Webtrends First Party Cookie and associated visit-related visit query parameters.
dcsOther	none	none	Reads customer provided cookie (specified in customerfpc property) and uses its value for unique visitor identification.
dcsTP	none	none	Tracks activity for SmartView using a meta tag. This function sets a cookie named WT_DC to identify SmartView pages. As a best practice, you should not modify any code in this function. For more information about using your JavaScript tag to track pages for SmartView, see the SmartView User's Guide.
dcsGetMeta	String containing meta tag content or null	name – String containing meta tag name	Reads a meta tag name and returns its content.
dcsAdSearch	none	none	Scans page for links containing ad click query parameters (specified in adclickparam property) and populates Webtrends Query Parameter for ad view (WT.ad).
dcsQP	String containing query parameter value or empty string	N – String containing query parameter name to search for	Search query string for a query parameter name and return its value.
dcsIsOnsite	Boolean indicating if domain name is onsite (true=onsite, false=not onsite)	host – String containing domain name to test	Compares domain name with list of onsite domains (specified in onsitedoms property) and determines if a match occurred.

dcsTypeMatch	Boolean indicating if a given filename type matches a list of types	pth – String containing filename to test typelist – String containing the list of file types	Extracts filename type and compares it with a list of filetypes and determines if a match occurred.
dcsEvt	Object containing click event	evt – Object containing event tag – String containing HTML tag name	Walks up the DOM hierarchy looking for a click target event with matching HTML tag name. Helper method for event handlers.
dcsNavigation	String containing element id or classname	evt – Object containing event	Walks up the DOM hierarchy looking for a matching HTML tag name (specified in navigation tag property). Helper method for event handlers.
dcsBind	none	event – String containing mouse event func – Object containing event handler function	Binds event handling function to mouse event.
dcsET	none	none	Binds event handling functions to mouse event.
dcsMultiTrack	none	String pairs containing query parameter key/ values to send to data collection servers	Helper method for globally available dcsMultiTrack function.
dcsCleanup	none	String pairs containing query parameter key/ values to retain	Clears out values in query parameter storage objects. Used in conjunction with dcsMultiTrack. String pairs passed as parameters are retained.
dcsSetProps	none	String pairs containing query parameter key/ values to retain	Helper method for dcsMultiTrack, dcsCleanup.
dcsSplit	Array of items	list - String containing comma separated list of items	Parses comma separated list of items into array and removes leading and trailing white space from each item.
dcsDownload	none	Object containing event	Download links click event handler.

dcsRightClick	none	Object containing event	Right click download links click event handler.
dcsDynamic	none	Object containing event	Dynamic links click event handler.
dcsFormButton	none	Object containing event	Form button click event handler.
dcsOffsite	none	Object containing event	Offsite links click event handler.
dcsAnchor	none	Object containing event	Anchor links click event handler.
dcsImageMap	none	Object containing event	Image map click event handler.
dcsMetaCap	none	none	Reads meta tags looking for matching names (specified in metanames property) and assigns content values to custom query parameters.
dcsEvi	none	none	Reads cookie (specified in evi.cookie property) and assigns value to query parameter (specified in evi.qp property). Optionally reads cookie crumb (specified in evi.crumb property) using separator (specified in evi.sep property).
dcsAdv	none	none	Wrapper method that dispatches several lower-level utility methods. Each utility method corresponds with a particular feature.
dcsVar	none	none	Gathers page data and assigns it to query parameter storage objects.
dcsEscape	String containing encoded value	S – String containing value to encode REL – RegEx object containing list of characters to encode	URL-encodes query parameter values.
dcsA	String containing query parameter key/ value	N – String containing query parameter key V – String containing query parameter value	Helper function used to compose data collection URL.

dcsEncode	String containing encoded query parameter value	S – String containing query parameter value	Helper function used to URL-encode query parameter values.
dcsCreateImage	none	dcsSrc – String containing data collection URL	Populates the Image array with the data collection URL. This causes the image request to occur.
dcsMeta	none	none	Reads meta tags looking for names containing WT, DCS, or DCSext. If found, the content values are assigned to corresponding query parameter storage objects. This method is called after dcsVar so that meta tag assignments tag precedence over all other query parameter assignments.
dcsTag	none	none	Composes data collection URL from query parameter storage objects and passes URL to dcsCreateImage method.
dcsCollect	none	none	High level function called to perform data collection request.
wtbind			Extends function object to maintain "this" object using a closure. Used by click event handlers.

Global Functions

In addition to the methods found in the Webtrends constructor function, the tag contains a few globally accessible functions. Access these functions through the global namespace using window. prefix rather than _tag. prefix.

Function	Return type	Parameter list	Description
dcsMultiTrack	none	String pairs that are query parameter key/values to send to data collection servers	Sends data collection request. Typically in response to an event.
dcsDebug	none	none	Creates popup window with data collection URL components.

URL Encoding

Certain characters can cause problems when used in query parameter values. For example, for a Webtrends query parameter assignment of `WT.ti="The Gettysburg Address";` Webtrends Data Collection servers write the following value to the log file:

```
&WT.ti=The Gettysburg Address
```

The space characters in this value cause problems because the space character is used to separate fields within a log file. The solution is to URL encode all query parameter values. URL encoding means replacing certain characters with their hexadecimal equivalents of the form %XX where % is the escaping character and XX is the character's numeric ASCII value. URL encoded characters are properly rendered in Webtrends reports. Continuing with this example, the URL-encoded form is as follows:

```
&WT.ti=The%20Gettysburg%20Address
```

Note that space characters have been replaced by %20. The tag URL encodes the following characters: tab, space, #, &, +, ?, ", \, and non-breaking spaces. These characters are defined in the regular expression list. The regular expression list contains regular expressions to search for, and the corresponding %XX replacement strings. Regular expression properties are used as arguments to the string.replace method. The tag URL encodes parameter values by passing them as arguments into the `dcsEscape` method.

Image Array

The tag uses an array of Image objects so that many hits to SDC from a single Web page can be made in rapid succession. The data collection request URL is assigned to the Image object's `src` property. This causes the browser to load the image into the image cache. This is a fast and efficient way to load the image that lends itself to repeated invocations of the tag.

Query Parameter Storage Objects

The following objects are properties of the Webtrends constructor function: DCS, WT, DCSext.

Each object contains query parameter mappings. Property name/value pairs correspond to the query parameter name/value pairs that are ultimately passed to Webtrends Data Collection servers. Each object serves as a container to group together like types of query parameters.

Object	Query Parameter Types Contained Therein
DCS	SDC-specific
WT	Webtrends Query Parameters
DCSext	User-defined

Adding customized information to the tag involves assigning property name/value pairs to the appropriate object. During object instantiation, empty objects are created. You can manually assign customized query parameters to the objects. For more information about adding custom query parameters, see "Adding Customized Information" on page 36. The `dcsMeta` and `dcsVar` methods automatically populate the objects with property name/value pairs. The `dcsTag` method enumerates values contained in each object, and composes the query parameter string that is passed to SDC.

SDC-Specific Query Parameters (DCS Object)

The DCS Object stores SDC-specific query parameters. The property names correspond to the following SDC-specific parameters: dcsqry, dcsref, dcsaut, dcsmet, dcssta, dcssip, dcspro, dcsbyt, dcsdat, dcsp3p, dcscfg, dcscip.

For example, by default the dcsref parameter is assigned as shown here:

```
DCS.dcsref=window.document.referrer;
```

This assignment is transformed into a query parameter as follows:

`&dcsref=`*encoded contents of* `DCS["dcsref"]`

For more information about using each of these query parameters, see .

Webtrends Query Parameters (WT Object)

The WT object stores Webtrends query parameters. The property names correspond to the type/attribute portion of the Webtrends query parameter.

For example, by default the WT.ti parameter is assigned this way:

```
WT.ti=document.title;
```

This is transformed into a query parameter as follows:

`&WT.ti=`*contents of* `WT["ti"]`

Note that `WT.` is added to the beginning of the object property name to create the query parameter key.

User-Defined Query Parameters (DCSext Object)

The DCSext object is used to store user-defined or "custom" query parameters.

For example, suppose you want to insert your own query parameter, apparel=jacket. This is how the DCSext.apparel parameter is assigned:

```
DCSext.apparel="jacket";
```

Alternatively, you can use the meta tag:

```
<meta name="DCSext.apparel" content="jacket">
```

which is transformed into the following query parameter:

```
&apparel=jacket
```

Be careful when naming user-defined query parameters. Do not use any of the parameter names used by SDC-specific parameters or by the Webtrends query parameters because those names are reserved.

Extended Internationalization Support

The JavaScript tag has extended internationalization support primarily for handling search phrases and query parameters in Asian languages. To enable this support, set the `i18n property true`. This support must be used in conjunction with Webtrends Conversion Plug-in. For more information, see "Internationalization and Webtrends" in the *Webtrends Administration User's Guide*.

Adding Customized Information

You can add customized query parameters by assigning property name/value pairs to the query parameter storage objects: DCS, WT, DCSext. As a best practice, you should add customized query parameters through HTML meta tags. You can do this quickly and easily using an HTML authoring tool. For more information about this process, see "Tagging Web Pages for SDC" chapter in *SmartSource Data Collector User's Guide*.

Alternatively, you can assign properties directly to the object in the inline HTML portion of the tag, beneath the "Add custom parameters here" comment. Note that in the case of a redundant parameter definition, an HTML meta tag assignment takes precedence over a direct object assignment. This permits override capability through HTML meta tags.

SDC-Specific Query Parameters

The JavaScript tag assigns several of the SDC-specific query parameters in the `dcsVar` method by default. Typically, you should not change these parameter assignments because they are required for base functionality. In a few cases, you can define additional SDC-specific parameters for added functionality.

The following example uses the `dcscfg=1` parameter to disable Webtrends Third Party Cookie.

You could use the following meta tag definition:

```
<meta name="DCS.dcscfg" content="1">
```

Alternatively, you could assign a value directly:

```
_tag.DCS.dcscfg="1";
```

Either approach results in the following query parameter definition:

```
&dcscfg=1
```

Webtrends Query Parameters

Webtrends Query Parameters are specially designed parameters that provide a powerful way to collect meaningful web data. Webtrends Query Parameters are passed in URLs through JavaScript to Webtrends Analytics for analysis. For more information, see .

By default, the Webtrends JavaScript tag defines several Webtrends query parameters. These

parameters are used by pre-configured custom reports in Webtrends. You can get additional insight by adding parameters recognized by the Webtrends Auto-Configuration feature.

For example, suppose you want a page to indicate that a shopper had just added a "widget" to the shopping cart. To use the Webtrends preconfigured "Shopping Cart" Scenario Analysis, you need to set the following parameters: `WT.pn_sku`, `WT.si_n`, `WT.si_p` or `WT.si_x`.

One way to pass these parameters is by using meta tags on your Web pages. In this example, the following meta tag definitions are used:

```
<meta name="WT.pn_sku" content="91x2G439bnM">
<meta name ="WT.si_n" content ="ShoppingCart">
<meta name ="WT.si_p" content T="CartAdd">
```

Alternatively, you could assign the parameters directly. In this case, the parameters are used as shown in the following example:

```
_tag.WT.pn_sku=91x2G439bnM;
_tag.WT.si_n=ShoppingCart;
_tag.WT.si_p=CartAdd;
```

Either approach results in the following query parameter definitions:

```
&WT.pn_sku=91x2G439bnM&WT.si_n=ShoppingCart&WT.si_p=CartAdd
```

User-Defined Query Parameters

You can also use custom (user-defined) query parameters. These query parameters are typically used in Webtrends custom reports as measures or dimensions. Be careful when naming custom query parameters. Do not use any of the parameter names used by SDC-specific parameters or by Webtrends query parameters because these names are reserved.

For example, to add a custom query parameter named `wtprod` and assign it the value Webtrends, you can use the following meta tag definition:

```
<meta name="DCSext.wtprod" content="Webtrends">
```

Alternatively, you could directly assign the parameter value as follows:

```
_tag.DCSext.wtprod="Webtrends";
```

Either approach results in the following query parameter definition:

```
&wtprod=Webtrends
```

Debugging Your Customizations

To debug the Webtrends JavaScript tag, there is no substitute for having a source level debugger. Microsoft Visual Studio for Internet Explorer and Venkman or Firebug for Firefox are both invaluable for setting breakpoints and inspecting variables. Browser plug-ins that permit sniffing of HTTP requests are also useful. HTTPWatch for Internet Explorer and HTTPFox or Firebug for Firefox can be handy. And finally, you may write JavaScript code to help inspect the data being sent to data collection servers. The following are a couple of ideas with regard to debugging.

Using dcsDebug

The tag contains a global function named `dcsDebug` that can be used to inspect the most recent data collection request. Invoke this function from the address bar of your browser as follows:

```
javascript:dcsDebug()
```

This will create a popup window that presents the data collection URL in an easily to read form. The URL is broken up into it’s constituent parts: protocol, domain, path, query parameters. Additionally, cookie information is presented.

Note that your browser may be configured to disable popup windows. If this is the case, the browser typically displays a warning, and asks if you would like to enable the popup. An affirmative response will display the debug information.

To perform simple debugging tasks, it is sometimes useful to use JavaScript itself to display the contents of variables.

The following examples contain some code snippets that are useful for customizing the tag.

Displaying the URL

Another way to look at the data collection URL is to a `document.write` statement inside the `dcsCreateImage` method as follows:

```
Webtrends.prototype.dcsCreateImage=function(dcsSrc){

   // Add the following line

   document.write(dcsSrc);


   if (document.images){

      this.images[this.index]=new Image();

      this.images[this.index].src=dcsSrc;

      this.index++;

      }

   else{

      document.write('<IMG ALT="" BORDER="0" NAME="DCSIMG" WIDTH="1" HEIGHT="1"
      SRC="'+dcsSrc+'">');

   }
}
```

This technique displays the URL directly into the Web page. Alternatively, you can use the JavaScript `alert` function instead of `document.write` to display the URL in an alert dialog.

Displaying the Query Parameter Storage Objects

The `dcsDebugObjects` function below shows a way to inspect the contents of the query parameter storage objects used by the tag:

```
function dcsDebugObjects(){
   if (typeof(_tag)!="undefined"){
      var DCS=_tag.DCS;
      var WT=_tag.WT;
      var DCSext=_tag.DCSext;
      var prop="";
      var data="";
      for (prop in DCS){
        if (DCS[prop]){
               data+="\nDCS."+prop+" = "+DCS[prop];
       }
      }
      for (prop in WT){
        if (WT[prop]){
               data+="\nWT."+prop+" = "+WT[prop];
       }
      }
      for (prop in DCSext){
        if (DCSext[prop]){
               data+="\nDCSext."+prop+" = "+DCSext[prop];
       }
      }
      window.alert(data);
   }
}
```

This function displays the contents of each object in an alert dialog. It is presented here to give an idea of how to access these important data structures in the tag.

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Chapter 5

Tracking Complex Web Page Interactions Using dcsMultiTrack

The Webtrends dcsMultiTrack function lets you log virtual page views to Webtrends SmartSource Data Collector (SDC) servers for nearly all types of events, including events within a Flash application as well as interactions triggering Ajax activity. Although the SDC tag captures page views automatically, not all relevant data is captured by the tag. dcsMultiTrack is a way to program a Web site page so that data captured using the dcsMultiTrack functions and parameters is recorded. Data that dcsMultiTrack can capture includes:

- Flash events such as start, finish, progress, and clicks within a Flash application
- Ajax interactions
- Downloads of files that can not be tagged, such as PDFs
- Form completion
- Any event that launches JavaScript

This document describes how to implement dcsMultiTrack API calls to track data that is not available through the advanced JavaScript tags. Advanced JavaScript tags are the primary source for tracking events, while dcsMultiTrack tags are used to implement tracking for events that are not available using the advanced tags. For information on using Advanced JavaScript tags, or for information about tracking Flash events, see "Customizing JavaScript Tags"in the *Webtrends Analytics Implementation and Maintenance Guide*.

How dcsMultiTrack Works

When a visitor performs an action that activates dcsMultiTrack, the function relies on the information saved when it was initially collected when the page was loaded. The information sent by the tag becomes a single log file line in the SDC log. Unmodified, this would duplicate a page view that was already recorded when the HTML page was initially loaded. However, dcsMultiTrack can overwrite or supplement any data you choose from the hosting page, enabling you to specify exactly what information is recorded in the hit being reported by dcsMultiTrack.

Additionally, when Web 2.0 technologies are implemented on a site that does not result in new HTML pages being loaded, such as Flash, Ajax, and other interactive elements, these interactions are not reported with the standard (non-dcsMultiTrack) implementation. When you implement dcsMultiTrack on a target page, data is collected each time an interaction occurs. User interactions, such as selecting different tabs or other options to change the content displayed, are reported in the same way as loading a new HTML page. Using dcsMultiTrack on sites that do not use Web 2.0 technologies provides a detailed view of how customers interact with the site when analyzing traffic.

Preparing for dcsMultiTrack

To implement dcsMultiTrack, you must include the dcsMultiTrack function in your Webtrends JavaScript tag. Because dcsMultiTrack relies on the standard Webtrends JavaScript tag to operate successfully, you must still deploy the standard Webtrends JavaScript tag on the HTML pages that include Flash, Ajax, or other technology you want to track using dcsMultiTrack.

If you use Webtrends Tag Builder to generate your tag, the JavaScript code includes the dcsMultiTrack definition. If you currently use an older version of Webtrends JavaScript code, you may need to modify the tag as described in the following sections. You can ensure that the version of the JavaScript includes the function definition for dcsMultiTrack by verifying that the line "`function dcsMultiTrack(){`" is present in the tag.

Adding the dcsMultiTrack Function to an Existing JavaScript Tag

If you are using an older version of Webtrends the JavaScript tag, and you do not want to deploy a new tag created with Webtrends Tag Builder, you can update your existing tag to enable dcsMultiTrack.

To add dcsMultiTrack to an existing tag:

1. Locate the dcsMultiTrack function.
   a. If you use the SmartSource Data Collector (SDC) with Webtrends software, this function is located in the `multitrack.js` file on the SDC server in the following directory:

      <*SDC Installation Directory*>\util\javascript\

   b. If you use Webtrends On Demand, or if you cannot locate the `multitrack.js` file, contact Webtrends Support to obtain a copy of the dcsMultiTrack function definition.
2. Open the `multitrack.js` file and select all the code in the file. Copy this text to the clipboard.
3. Paste this code into your Webtrends Javascript tag immediately after the comment `Add customizations here` as shown in the following Webtrends JavaScript tag excerpt:

```
// Add customizations here

function dcsVar(){
var dCurrent=new Date();
WT.tz=(dCurrent.getTimezoneOffset()/60*-1)||"0";
WT.bh=dCurrent.getHours()||"0";
```

Using dcsMultiTrack

Any number of parameters can be passed to dcsMultiTrack provided they are passed to the function in pairs, and the parameters used are JavaScript strings enclosed in single or double quotes. The first element of each pair is the name of the Webtrends query parameter you want to set, while the second element of each pair is the value you want the Webtrends query parameter to contain. To be recognized by Webtrends query parameters, the query parameter names must begin with `WT.`, `DCS.`, or `DCSext.`. The values that you specify replace the values that would normally be gathered by the Webtrends JavaScript tag. To avoid duplication of page views, you can report the dcsMultiTrack event with a different URL and page title than the initial page view for the HTML page. This requires using the `DCS.dcsuri` and `WT.ti` parameters with the dcsMultiTrack function. The following example illustrates this implementation:

```
dcsMultiTrack('DCS.dcsuri','/folder/movie.swf','WT.ti',
   'Name%20of%20the%20Movie');
```

In the above example, `%20` was used to replace spaces in the page title. Because spaces can have a special meaning in many scripting languages, the best practice is to use URL encoding to represent any non-ASCII character, especially spaces, quotes, and parentheses.

Other parameters may be passed to dcsMultiTrack as well. Any Webtrends query parameter that you can set with a META tag can also be set with dcsMultiTrack. If the page contains META tags or parameters that specify Webtrends query parameters that data is sent to the SmartSource Data Collector along with values for the initial HTML page view. For a complete list of Webtrends query parameters, see the *Webtrends Administrators User's Guide*. For a list of JavaScript event handlers and their corresponding attributes, refer to a JavaScript Web site such as http://www.w3schools.com/jsref/jsref_events.asp.

In the following example, values from the initial HTML page view are sent to the Webtrends data collectors. If the page contains META tags or parameters that specify Webtrends query parameters, that data is also sent.

To invoke the dcsMultiTrack function to report a virtual page view to the Webtrends data collector, use the following syntax:

```
dcsMultiTrack('parameter1', 'value1', 'parameter2', 'value2');
```

Persistent and Inherited Query Parameters

If you do not specify values for each parameter-value pair used with the dcsMultiTrack call, Webtrends uses values from the most recent dcsMultitrack call. These inherited parameters are specific to dcsMultiTrack while JavaScript query parameters are persistent.

If you want to exclude data from specific query parameter tags, you can do so by specifying blank values in the dcsMultiTrack statement. For example, to remove the `WT.mc_id` campaign tracking parameter, modify the dcsMultiTrack tag syntax as follows:

```
dcsMultiTrack('DCS.dcsuri','/folder/movie.swf','WT.ti',
   'Name%20of%20the%20Movie', 'WT.mc_id', '');
```

Tracking Off-Site Links and Downloads

If you want to track a specific HTML link, whether to another site, a downloadable file, or any other linked target, you can implement the dcsMultiTrack function to collect this data whenever the link is clicked, using the JavaScript "onclick" event. The following example illustrates how to track clicks on a link that leads to a different site by activating the dcsMultiTrack function:

```
<a href="http://www.somedomain.com/folder/page_of_interest.html"
onclick="dcsMultiTrack('DCS.dcssip', 'www.somedomain.com',
'DCS.dcsuri', '/folder/page_of_interest.html',
'WT.ti', 'Page%20of%20Interest');">
Page of Interest</a>
```

In this example, the URL `http://www.somedomain.com/folder/page_of_interest.html` is included in the Pages report, as well as other reports. Even though the target page is not tagged with the Webtrends JavaScript code, the dcsMultiTrack function call simulates that page view by setting the domain, URL stem, and page title.

The same implementation can be used to track click frequency for link to download a file, such as a PDF file, Word document, sound file, or any other resource links included on the HTML page. The following example illustrates how the dcsMultiTrack function is used to report clicks to a PDF document:

```
<a href="/downloads/sales_brochure.pdf"
onclick="dcsMultiTrack('DCS.dcsuri', '/downloads/sales_brochure.pdf',
'WT.ti', 'Sales%20Brochure');">
Download a Sales Brochure</a>
```

Using this code would return reports showing a URL at the same domain as the HTML page, with the file path `/downloads/sales_brochure.pdf`. If you have configured Webtrends to count files with extension “PDF” as downloads, then this appears in the Downloaded Files report. Depending on your page view determination configuration, this may also appear in the Pages report. For more information, see the section “Tracking Page Views and Downloads” in the chapter *Refining Analysis* in the *Webtrends Administration User’s Guide*.

use the software version of Webtrends Analytics with the default configuration, this appears in the Downloaded Files report, but not the Pages report. Webtrends Analytics On Demand includes this information in both reports.

It is possible to use general event handlers to track all links or all downloads for a page or site. However, this level of implementation is beyond the scope of this document. Before implementing dcsMultiTrack to track a specific link or download, check with your Webtrends administrator to verify if such an event handler is implemented on your site.

Tracking Form Elements

The dcsMultiTrack function can be activated by any JavaScript event. For example, if you want to track how a visitor progresses through a particular form so that you can configure a Scenario Analysis report to track where visitors drop out of the process, you can use the dcsMultiTrack function to track each form element separately, sending a virtual page view whenever a visitor fills out a form element.

The following example illustrates how to set up dcsMultiTrack to track which form elements the visitor completes. Note that each form element to be tracked is tagged with `DCS.dcsuri` and specifies the URL page where the tagged element `.htm` file resides.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<title>Sign Up Here!</title>
<meta name="WT.si_n" content="RegistrationForm">
<meta name="WT.si_x" content="1">
</head>
<body>
<form action="http://www.tipjar.com/cgi-bin/test method=post>
<p>
<input name="First Name" type="text" value="" size="40" maxlength="100"
onchange="dcsMultiTrack('DCS.dcsuri', '/FirstName.htm',
'WT.ti', 'First%20Name%20Field', 'WT.si_n', 'RegistrationForm',
'WT.si_x', '2');">First Name</p>
<p>
<input name="Last Name" type="text" value="" size="40" maxlength="100"
onchange="dcsMultiTrack('DCS.dcsuri', '/LastName.htm',
'WT.ti', 'Last%20Name%20Field', 'WT.si_n', 'RegistrationForm',
'WT.si_x', '3');">Last Name</p>
<p>
<input name="Email Address" type="text" value="" size="40" maxlength="100"
onchange="dcsMultiTrack('DCS.dcsuri', '/Email.htm',
'WT.ti', 'Email%20Field', 'WT.si_n', 'RegistrationForm',
'WT.si_x', '4');">Email Address</p>
<p>
<input type=submit value="submit"></p>
</form>
</body>
</html>
```

When a visitor fills in each field of the form, the Pages report shows the URL for the HTML page itself, as well as URLs for `/FirstName.htm`, `/LastName.htm`, and `/Email.htm`. The Registration Form Scenario Analysis report is also be populated with all four steps reflected in this example. If the "Thank you" page has the appropriate META tags for the fifth step of the scenario, you can obtain a granular view of how visitors are interacting with your form.

Tracking Ajax Interactions

The term Ajax is used to refer to a variety of specific combinations of technologies. Most of these technologies focus on the use of JavaScript to directly modify the Web page using a Document Object Model (DOM), based on customer interaction and data fetched from the server. Ajax can be used to provide rich applications within the Web browser ranging from calendar applications to mapping tools, and can also be applied in a simpler form for minor changes like updating a drop-down menu based on visitor input.

Because Ajax typically uses JavaScript events, implementing dcsMultiTrack within an Ajax application is straightforward. If an Ajax event is triggered by a customer interaction that you want to report on, the same Ajax event that updates the page in response to that interaction can also call the dcsMultiTrack function.

The following example illustrates a simple Ajax control implementation where the visitor's selection in a drop-down menu changes the options available in a second drop-down menu. The dcsMultiTrack function tracks each time a user chooses a different option in the first drop-down box.

```
<select name="drop-down1" onChange="UpdateDropDown(this.value);">
<-- options -->
</select>
<select name="drop-down2">
</select>
...
<script type="text/javascript">
function UpdateDrownDown(OptChoice) {
   var drop2 = parelmts["drop-down2"];
// fetch data for the second dropdown
parelmts.length = xmlreply.length;
   for (o=1; o < xmlreply.length; o++) {
      drop2[o].text = xmlreply[o];
   }
   dcsMultiTrack('DCS.dcsuri', '/dropdown.html', 'WT.ti', 'Drop%20Down',
      'DCSext.Choice', OptChoice);
}
```

In this example, each use of the first drop-down menu is reported as a virtual page view at the URL `http://www.yourdomain.com/dropdown.html`. The user-defined Choice parameter is also populated with the selection the visitor made in the drop-down menu.

Due to the wide variety of Ajax implementations, attempting to provide examples for every possible scenario is beyond the scope of this document. However, the examples provided are applicable to nearly any Ajax application. In most cases, the modification to the DOM is encapsulated within a function. Therefore, activating the dcsMultiTrack function within your own function is a discreet method to implement Webtrends tracking within your Ajax application.

Tracking Using Document Object Model (DOM)

For complex Web sites that enforce rigorous separation of logic and presentation, you can use the Document Object Model (DOM) method to implement the dcsMultiTrack function. Implement dcsMultiTrack on a DOM element ID or other criteria such as anchor tags that lead to PDF documents. To illustrate basic implementation using DOM with dcsMultiTrack, two examples are included in the following sections. The first example illustrates activating dcsMultiTrack for a particular DOM element ID. The second example illustrates activating dcsMultiTrack for a specific DOM tag name.

Note

Using DOM methods with dcsMultiTrack is a complex implementation that requires an understanding of DOM methodology and programming skills that are beyond the scope of this document. The following examples are for illustration purposes only–**do not copy and paste the code examples to your Web page** as they may not work as expected.

For information on DOM methods, code samples, and validating syntax, see http://w3schools.com/htmldom/default.asp or other applicable Document Object Model Web sites.

Example: Specifying DOM Element ID with dcsMultiTrack

The following example shows how to activate dcsMultiTrack for a particular DOM element ID:

```
<script type="text/javascript">
var x=getElementByID("page_of_interest");
x.onclick="dcsMultiTrack('DCS.dcssip', 'www.domain_name',
 'DCS.dcsuri', '/folder/page_of_interest.html',
 'WT.ti', 'Page%20of%20Interest');"
</script>
...
<a id="page_of_interest" href="http://www.somedomain.com/folder/
page_of_interest.html">Page of Interest</a>
```

Example: Specifying DOM Tag Name with dcsMultiTrack

The following example shows how to activate dcsMultiTrack for a particular DOM tag name:

```
<a id="page1" href="http://www.somedomain.com/folder/page_of_interest.pdf">Page of
Interest 1</a>

<script type="text/javascript">
var x=document.body.getElementsByTagName("a");
for (i=0;i<x.length;i++) {
   if (x[i].href.toUpperCase().indexOf("PDF") >= 0) {
      var url_split=x[i].href.match(/(\w+):\/\/([\w.]+)(\/\S*)/);
      var wt_domain=url_split[2];
      var wt_uri=url_split[3];
      var wt_title=x[i].innerHTML
      x[i].onclick=function(){
       dcsMultiTrack('dcssip', wt_domain, 'dcsuri', wt_uri, 'WT.ti', wt_title);
      };
   }
}
</script>
```

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate branding updates.
v8.7	March, 2009	Added Document Revision History section.

Chapter 6

Creating Webtrends Analytics Profiles

This chapter describes the kind of information stored in report profiles and provides instructions for creating a profile using the Basic and Advanced profile wizards. For more information about using each dialog in the profile wizards, see the Administration Help.

How Profiles Work

Profiles specify all of the information needed to generate reports from a web data file. They define the location of your web server data and how it should be analyzed. For example, profiles can specify information such as:

- The type of web data Webtrends analyzes and where to find it
- The platform type that the profile tracks, such as a site or mobile app.
- Whether your web site resides on a single server or on multiple servers
- The location of your home page
- Whether to apply data filters
- Which users can access the profile
- Which reports to create
- When to update reports

Each profile is associated with a set of log files from which it draws data and one or more templates, which determine the set of reports that can be rendered from the analyzed data. When you analyze a profile, Webtrends creates a set of Report databases. You can use Webtrends Analytics Reports to view reports for a specified profile and template based on the data in the Report databases.

Profile Creation

To begin creating a profile:

1. In the left pane, click **Administration > Web Analysis > Reports & Profiles**.
2. Click **New**.

Using the Profile Wizard

Webtrends allows you to create profiles in two modes. In the Basic mode of profile creation, Webtrends uses the most common settings to produce reports quickly with minimum configuration. The Basic mode creates a Standard Full-Featured Analysis profile. You can also use the Advanced profile settings to customize your profile by providing more information about your web data, your site configuration, and your reporting preferences. Advanced profile options are available for site profiles only.

To create a profile using the Basic wizard:

1. In the Profile Name dialog, enter a name for your profile.
2. If you are a Webtrends On Demand user, specify the time zone you want to use when displaying your reports.
3. Select a profile type for the platform you want to track: Site, iOS App, BlackBerry App, Android App, Windows Phone App, Mobile App.
4. Enter the home page of your web site or the name of your app. The domain or name you provide here maps to a space in Analytics 10. When you enter an existing domain or app name, it maps to an existing space; a new domain or app name creates a new space.
5. In the Data Sources dialog, specify a data source (if you have already configured one) or create a new data source. A data source identifies the location of the web data that you want to track with this profile.
6. In the Session Tracking dialog, specify how to identify user sessions for the profile.
7. In the Summary dialog, review your settings.

Advanced Profile Settings

Use the Advanced Profile settings if you want to configure special Webtrends features such as:

- Advanced SmartView reports
- Report Packs
- Profile analysis scheduling (Webtrends software users only)
- Advanced reporting features including:
  - Campaign and Scenario Analysis
  - Content Group Analysis
  - Hit and Visit Filters
  - URL Parameter Analysis
  - URL Search and Replace

Click **Additional Settings** in the last wizard dialog to access these advanced settings. Advanced profile options are available for site profiles only.

You should also use Advanced Profile settings if you want to create special types of profiles such as Parent-Child profiles. For more information about configuring these specialized profile types, see the Help and the *Webtrends Administration User's Guide*.

To create a profile using the Advanced Profile settings:

1. In the Profile Name dialog, enter a name for your profile.
2. If you are a Webtrends On Demand user, specify the time zone you want to use when displaying your reports.
3. Select **Site** as the profile type.
4. Enter a web site domain name. The domain you provide here maps to a space in Analytics 10. When you enter an existing domain, it maps to an existing space; a new domain creates a new space.
5. Select the **Advanced profile options** check box. You see the Advanced Profile Options dialogs in the wizard listed in the left column.
6. Click **Next** to complete the dialogs. For detailed information about each setting, click the **Help** icon. To configure more profile options than the wizard provides, click **Additional Settings** in the Summary dialog.

Advanced Profile Dialogs

The following dialogs are included in the Advanced Profile wizard. For detailed information about how to complete each dialog, see the Help. Dialogs marked with an asterisk (*****) may not be displayed depending on the wizard settings you choose. However, choosing Advanced Settings will typically display all the dialogs for the Advanced Profile wizard.

Data Sources
Specifies the location of the web data file.

General
Specifies the portion of the log to analyze, whether to retrieve HTML page titles, whether to enable Express Analysis, and time zone behavior. (If you are a Webtrends On Demand user, specifies the time zone to be displayed when displaying reports).

Home*
Specifies the location of the site home page.

Host Binding*
Specifies whether analysis should run only on certain computers or groups of computers.

Page File Types*
Specifies which file types Webtrends counts as page views.

Parent Child*
Specifies the settings used to create a Parent-Child profile.

Post-Analysis*
Specifies whether any programs should run immediately after analysis.

Pre-Analysis*
Specifies whether any programs should run immediately before analysis.

Profile Class
Specifies the type of web data to use and whether to create a standard, Advanced SmartView, Event Database, or Parent-Child report.

Profile Name
Specifies the name of the profile, the site domain name (used to identify the SmartView domain and the web site URL) and (for Webtrends On Demand users) the time zone.

Schedule
Specifies when analysis should occur.

Session Tracking*
Specifies whether to apply profile-specific settings for tracking user sessions.

Site Configuration*
Specifies whether the web server resides on one server or multiple servers.

SmartView*
Specifies whether to enable SmartView reporting. For advanced SmartView profiles, specifies whether to analyze the home page domain or another domain.

Summary
Specifies the settings established for the profile.

Report Packs
Specifies the licensed report packs you want to use with the current profile. Report Packs determine what kinds of reports Webtrends creates, and thus the type of data included in reports for this profile.

URL Rebuilding*
Specifies whether to apply settings that modify URLs before analysis to provide more accurate reporting.

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Spring 2011 Release	April, 2011	Updated profiles for Analytics 10; included profile type; moved Report Packs to Advanced Options.
Fall 2009 Release	November, 2009	Corporate branding updates.
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Chapter 7

Setting Up Webtrends Users and Roles

Webtrends provides a highly configurable array of user rights and roles to help you set up user access that reflects your organizational security model. Depending on your organizational structure, you may find that you want to provide varying levels of access to different features and objects within Webtrends.

This chapter describes how to create users and assign user rights to various features of Webtrends. At the most basic level, creating a user grants access to a Webtrends login. However, by selecting specific rights for each user, you can define very specific and detailed user rights according to each user's job function and data access requirements. If you have more than a handful of Webtrends users, you can reduce configuration time using groups of rights called *roles*. Instead of repeatedly defining many individual rights for each user, you can streamline rights assignment by assigning each new or existing user a role.

Understanding Webtrends User Rights

When you configure rights for a user or role, Webtrends provides three different types of rights: action rights, profile rights, and template rights. Selecting these rights defines the permissions granted to a user or role. Action rights are designed to control access to functionality. Profile and template rights provide access to the data and reports contained in specific profiles and templates.

Webtrends user rights are cumulative, and greater rights grant implied access to lesser ones. For example, granting Create rights also grants View, Edit, and Delete rights. Similarly, if you have only View rights to all profiles through the Action Rights dialog, you can still be granted Edit rights to any specific profile in the Profile Rights dialog.

Understanding Action Rights

Action rights are Create, View, Edit, and Delete rights to specific Webtrends features and functionality. (These rights were known as User Rights in previous versions.)

Action rights can provide different levels of functionality to users or groups of users. For example, users in your organization who primarily use Webtrends to view reports or schedule report exports probably do not need rights to manage users, set system options, and perform administrative functions. By providing users only with the rights they need, you can ensure that your configuration is secure and your report users do not become confused by too many configuration options.

Understanding Profile and Template Rights

Profile and Template rights are Create, View, Edit, and Delete rights to individual profiles and templates within Webtrends.

Profile and Template rights can provide better security and more targeted report content. In cases where different users and roles should not see the same data, you can use these rights to determine which profile data users can access and the number and arrangements of the reports they see. For example, Webtrends users who provide third-party reporting services often use profile rights to make sure each company only has access to its own data.

Similarly, you can limit template rights to make only certain groups of reports visible to each user or role. For example, if you only want to provide Search reports to your Search Marketing group, you can create a template that contains only those reports and grant template rights to each user in the group. Alternately, you can create a Search Marketing role that has those rights and assign it to all users in the group.

Note

Granting access to all profiles or all templates listed in the Profiles Rights and Template Rights dialogs does not also grant access to newly created profiles and templates. To grant rights to all current and future templates and profiles, use the Profiles and Templates rights in the Action Rights dialog.

Understanding Webtrends User Roles

User roles are predefined sets of user rights that you can apply to more than one user. If you have multiple users who need the same permissions, you can avoid recreating the same rights for each user by creating a role. Then assign the same role to each user by selecting the role in the General dialog of the User settings.

Configuring User Roles

To create a user role:

1. In the left pane, click **Administration > Application Settings > User Management > Roles.** A list of the current users roles opens.
2. Click **New**.
3. Specify a name for the role. For example, you could name a role based on a job function such as Accounting, a third-party company name, or a role within Webtrends such as “Report Scheduler.” Click **Next**.
4. In the Action Rights dialog, set rights to functionality for this user. For more information about Action rights, see “Understanding Action Rights” on page 53. For more information about individual Action rights, see Help. Click **Next**.
5. In the Profile Rights dialog, set rights to specific report profiles for this user. For more information, see “Understanding Profile and Template Rights” on page 54.
6. In the Template Rights dialog, set rights to specific report templates for this user. For more information, see “Understanding Profile and Template Rights” on page 54.

Applying Preconfigured Roles to Existing Users

User roles overwrite any existing rights for each user. If you have existing users and want to migrate them to use roles, you can edit each user and select a role in the General dialog.

Caution

After you assign a role to an existing user and click Save, you cannot recover any previous user rights settings associated with that user. Because Webtrends On Demand used a different method to assign user rights in previous releases, Webtrends On Demand administrators should plan carefully before mapping and applying roles.

Adding Users

Webtrends On Demand administrators create users and perform some other administrative tasks in On Demand Accounts. For help creating users, see the *Webtrends On Demand Accounts User Guide* or the Webtrends On Demand Accounts Help.

Tip

If you plan to create customized user roles and assign them to the users you create, configuring roles first will save you configuration time. For more information about roles, see "Understanding Webtrends User Roles" on page 54.

About View Only Permissions

For all users who have no other permissions than View Reports, make sure the user has view rights assigned in the template. Otherwise, the user with View Reports only rights will not be ale to view any reports if they are not listed in the template. Users with View Reports rights have access to Webtrends Analytics Reports only.

Note

Users who have View Only permissions are also automatically granted access to view profiles. However, because of the extreme limitation of View Only permissions, viewing profiles is not enabled for these users.

The *Webtrends Analytics Report User's Guide* is available for those using Webtrends Analytics Reports. You can download this guide from the Webtrends Customer Center.

Predefined Role Settings

The following graphic shows the settings for each of the predefined user roles included in Webtrends. You can use this comparison to determine whether the predefined roles are appropriate for your users. Settings marked N/A are not available rights for a specific feature or functionality.

Permission	Create			Delete			Edit			View		
	Administrator	Report Manager	Report User	Administrator	Report Manager	Report User	Administrator	Report Manager	Report User	Administrator	Report Manager	Report User
Accounts (WTOD only)	1	0	0	1	0	0	1	0	0	1	1	1
Analysis Options	1	1	0	1	1	0	1	1	0	1	1	0
Application Settings (Software Only)	1	0	0	1	0	0	1	0	0	1	0	0
Custom Reports	1	1	0	1	1	0	1	1	0	1	1	1
Custom Reports in Advanced Mode	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0	N/A	N/A	N/A
Data Sources	1	0	0	1	0	0	1	1	0	1	1	0
Event Database Free Form Query Access (Software Only)	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	0	0
Event Database Full Named Query Access	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	0	0
Export to Database	1	1	1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Export Word , CSV, PDF Reports	1	1	1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Goals	1	1	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Log File Import (WTOD only)	1	0	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Log File Download (WTOD only)	1	0	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Marketing Warehouse Configuration	N/A	N/A	N/A	N/A	N/A	N/A	1	0	0	N/A	N/A	N/A
Marketing Warehouse Profiles	1	0	0	1	0	0	1	1	0	1	1	0
ODBC Driver	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0
Profiles	1	0	0	1	0	0	1	1	0	1	1	1
Report Configuration	1	1	0	1	1	0	1	1	0	1	1	0
Report Designer	1	1	0	1	1	0	1	1	0	1	1	0
Reports	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1
Scheduled Export To Database	1	1	1	1	1	0	1	1	0	1	1	1
Scheduled Word, CSV, PDF, Reports	1	1	1	1	1	0	1	1	0	1	1	1
Scheduler (WT Software only)	1	0	0	1	0	0	1	0	0	1	0	0
Score	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1
Score Rule Set Configuration	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0	N/A	N/A	N/A
Score Rule Set Enablement	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0	N/A	N/A	N/A
Segment	1	1	0	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1
Shared Bookmarks	1	1	1	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
SmartView	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1
Templates	1	1	0	1	1	0	1	1	0	1	1	0
Translation File Import (WTOD only)	1	1	0	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Users	1	0	0	1	0	0	1	0	0	1	1	1
Visitor History Export	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0
Visitor Intelligence (Warehouse Reports)	1	1	0	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1
Web Services	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	0
WebTrends Explore	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	1	1	1

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate branding updates.
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Chapter 8

Tracking Visitor Sessions

In order to provide the most insightful data, Webtrends requires that you use a strong method of identifying visitors. For Webtrends Analytics, strong identification methods are cookies and authenticated user IDs. For Webtrends Visitor Data Mart, cookies are required to track visitor-related data. Because cookies are the recommended best practice, this chapter focuses on cookies. It explains the differences between first-party cookies and third-party cookies and describes how you can use cookies to track visitor sessions. For information about alternative methods of identifying visitors for session tracking, see "Visitor Identification" in the *Webtrends Guide to Web Analytics*.

This chapter assumes that you use Webtrends Analytics On Demand or have installed Webtrends SmartSource Data Collector (SDC) and have configured an SDC site map. For more information about installing SDC and configuring an SDC site map, see the "SmartSource Data Collector Installation" in the *Webtrends SmartSource Data Collector User's Guide*. If you do not intend to use first-party cookies with SDC, see "Using First-Party Cookies Without SDC" on page 69.

Note
This chapter does not apply to Webtrends Analytics On Demand Small Business.

What is a Cookie?

A cookie is a piece of identifying data, typically created by a web server. A web site sends a cookie to a visitor's browser and stores it on a visitor's computer either temporarily (for that visit session only) or permanently on the hard disk (or until the visitor deletes them). Temporarily stored cookies are called session cookies. Cookies stored on the hard disk are called persistent cookies.

Persistent cookies can identify a visitor as a new or returning visitor by storing a value that uniquely identifies each visitor. If a visitor has been to the site before, a cookie is sent to the web server with the request for a particular page. The web server checks for the presence of a cookie in the request and if no cookie is detected, the web server generates the cookie and sends it with the response to the visitor's browser. When the visitor returns, the cookie is included in the request, the web server detects the cookie and recognizes the visitor as a returning visitor. The web server then writes the cookie to the log file in the `cs(Cookie)` field.

Why Web Browsers Reject Cookies

Whether web browsers are likely to accept a cookie strongly depends on whether the cookie is a first-party cookie or a third-party cookie.

A cookie served from a domain other than the domain that your visitor requests from your web site is considered a *third-party cookie*. Webtrends On Demand and SmartSource Data Collector (SDC) have historically used cookies as the primary method to obtain visitor information.

Tracking visitors accurately is paramount for confidence in your web analytics results. Studies by leading analyst research firms such as Jupiter Research and Forrester have indicated that increasingly high-rates of cookie rejection and deletion by Internet users makes third-party cookies an unreliable method for collecting and reporting on web marketing results. In fact, Jupiter currently believes third-party cookie rejection rates are as high as 28%.

In response, Webtrends conducted its own research, analyzing third-party cookie rejection rates for 5 billion visitor sessions between January 2004 and April 2005. Webtrends research found cookie rejection rates to be somewhat lower but still significant, revealing that on average 12% of Internet user traffic is blocking or preventing third-party cookies from being set on computers, and that this trend can be as high as 17% for some vertical industries, such as retail. In addition, when analyzing the third-party cookie rejection trend since the beginning of 2004, Webtrends findings show that third-party cookie rejection has increased by a factor of four to its current rate.

Internet users commonly reject third-party cookies as part of their security measures. Some of the most common reasons for the increase in the rejection of third-party cookies are:

- Anti-spyware programs are designed to remove cookies that surreptitiously monitors visitors' web activities. These programs often consider hosted web analytics services to be spyware and thus target their cookies for removal from your visitors' computers.
- Current browser technologies such as Microsoft's Internet Explorer and Mozilla's Firefox make it easier for visitors to reject third-party cookies.

Negative Impact of Third-Party Cookie Rejection

Cookie-dependent analytics solutions rely on the cookie as the method to identify one unique browsing session from another. There are a number of business issues that arise from third-party cookies being rejected or deleted on a regular basis:

- **Inaccurate Visitor Metrics**: At its most fundamental level, if an Internet user has configured the browser security settings to automatically reject third-party cookies, that visitor will not be properly counted in your web analytics results. As mentioned earlier, market estimates project this to be anywhere from 12% to 28% of Internet users on average.
- **Deceiving Retention Based Metrics**: Taking this one step further, if "John Doe" visits your web site on May 15 and accepts the third-party cookie, he will be recognized as a new visitor. If John then deletes all of his third-party cookies with his anti-spyware application on May 16th and returns to the site on May 17th, the analytics solution will identify John as a new visitor, since he no longer has the cookie on his computer. This would have an impact of under representing your retention based metrics such as your repeat visitor rate.
- **Inaccurate Conversion Metrics**: Cookie deletion also has an impact on your conversion rate for new visitors versus repeat visitors.

Conversion rate = (conversion actions taken/number of visitors) X 100

where a conversion action is an action indicating visitor conversion, such as an order, and the visitors may be new or repeat visitors.

As pointed out in example #2 (directly above), if the cookie is being systematically deleted, repeat visitor rates are going to be under-counted and new visitor rates are going to be over-counted, skewing your conversion rate metric by which you analyze your site's overall effectiveness.

- **Unreliable Campaign, Search and Merchandising Reports**: In addition to tracking the behavior of a visitor to the site in general, many analytics providers correlate visitor response and site interaction to a specific campaign, search engine or product in an attempt to understand precisely which campaign or merchandising offer inspired the Internet user to take an action; much of this information can rely on information stored in the cookie. If the cookie is rejected or deleted from the Internet user's browser, reports designed to identify latent or deferred conversion to a campaign or merchandising offer will be misrepresented. It is also important to note that the longer that you track conversion to an individual marketing activity, the more likely it is that your metrics are inaccurate, as the likelihood the user deletes the third-party cookie increases.

Solving Rejection with First-Party Cookies

For most business models, *first-party cookies* are regarded as the most reliable method to measure visitor activity. A cookie served directly to your visitors by your own web server is registered by the browser is a first-party cookie. Whereas a third-party cookie is set by the analytics vendor, an entity with which the web site visitor does not have a relationship, the first-party cookie is set by the business or organization with which the Internet user has specifically chosen to do business. Because of this relationship, the first-party cookie is deemed a more secure cookie by the user.

First-party cookies are considered less of a security risk than third-party cookies and are more likely to be accepted by the browser. By issuing first-party cookies, your benefits include:

- Most accurate visitor metrics
- Compatibility with data collected from existing Webtrends data sources that used Webtrends third-party cookies

If you use Webtrends Analytics software, you may already have a method of setting first-party cookies. If you use Webtrends Analytics On Demand or Webtrends Analytics software with SDC, your Webtrends JavaScript tag is configured to use first-party cookies by default. For more information, see “Using the JavaScript Tag to Track Cookies” on page 63.

How Webtrends Uses Cookies

As long as the visitor’s browser accepts cookies and the visitor does not delete the cookie from the computer, Webtrends can use the cookie to determine whether the visitor is a returning visitor or a first-time visitor. Webtrends can also use cookies to strongly identify visitors and use this information to develop a rich repository of visitor history which you can use for reporting.

Methods for Generating First-Party Cookies

You can generate first-party cookies that Webtrends can use to track visitor sessions using one of the following methods:

- Allow the Webtrends JavaScript tag to serve cookies
- Configure your web server to serve cookies
- Use the Webtrends Cookie Plug-in to serve cookies

This section discusses each of these methods.

Using the Webtrends JavaScript Tag

If you use Webtrends On Demand or Webtrends software with SDC, your best choice for generating first-party cookies is by allowing your JavaScript tag to generate them. By default, the JavaScript tag generates the first-party cookie and passes it in the query string as the `WT.co_f` query parameter. With this method, you do not need to configure your web server to generate cookies.

Using Your Web Server to Generate Cookies

Most modern web servers contain functionality for serving cookies. If your web server is already configured to serve cookies, you should use this cookie to identify your visitors. This method is suitable whether you use Webtrends software or Webtrends On Demand.

This section describes how some commonly used web servers deliver cookies.

Apache Web Server

Apache provides the `mod_usertrack` module for click stream logging of visitor activity on a site. `Mod_usertrack` sets a cookie with a unique identifier. Enable `mod_usertrack.so` by adding this dynamically shared object to the LoadModule list, and setting the “CookieTracking on” directive in the `httpd.conf` file. The following directives provide additional control: CookieDomain, CookieExpires, CookieName, CookieStyle.

For more information, see `www.apache.org`.

Microsoft Internet Information Server (IIS)

Active Server Pages

Microsoft ASP supports the notion of a session management through the Session object. Session keys are stored in the ASPSESSIONID cookie.

As an alternative, you can manage your own tracking cookie using the Response.Cookies Collection of the Response and Request objects.

For more information, see `www.microsoft.com`.

Site Server

Microsoft Site Server includes a User identification Filter (ISAPI filter) called `mss_log.dll`. This filter generates a 32-byte GUID that is stored in the `SITESERVER` cookie.

For more information, see `www.microsoft.com`.

iPlanet/SunOne

iPlanet/SunOne's servlet engine supports Java Server Pages. You can manage your own tracking cookie by using the Cookie class and the `HttpServletResponse.addCookie` method, and the `HttpServletRequest.getCookies` method.

For more information, see `www.java.sun.com`.

- Modifying the Webtrends JavaScript tag to serve cookies
- Modifying the Webtrends JavaScript tag to serve cookies

Using the Webtrends Cookie Plug-in

The Webtrends Cookie Plug-in is software that you can install on your web server to generate first-party cookies. If your web server cannot be configured to serve cookies, and you use Webtrends software without SDC, this is your best choice for generating first-party cookies.

The Cookie Plug-in supports Apache, Microsoft IIS, and iPlanet/SunOne web servers. For more information about installing and using the Cookie Plug-in, see the *Cookie Plug-in User's Guide*.

Configuring Webtrends for First-Party Cookie Tracking

If you use Webtrends On Demand or Webtrends software with SmartSource Data Collector, you can configure your JavaScript tag to recognize the first-party cookie method that you use. By default, the JavaScript tag generates the first-party cookie for you. For more information, see "Using the JavaScript Tag to Track Cookies" on page 63.

If you use Webtrends software without SmartSource Data Collector, you simply need to create a Session Tracking definition configured to use your cookie.

Using the JavaScript Tag to Track Cookies

If you use Webtrends On Demand or Webtrends software with SmartSource Data Collector, the JavaScript tag is configured to track first-party cookies by default. Use the Tag Builder at tagbuilder.webtrends.com to create your tag and specify any additional cookie settings. The following procedure has four main steps.

Use the following steps in sequence to configure first-party cookie tracking.

1. Implement the JavaScript tag you created using the Tag Builder. See "Implementing the JavaScript Tag" on page 64.

2. Edit a profile and specify the first-party cookie data source. See "Specifying the First-Party Cookie Data Source" on page 64.
3. For the same profile, specify session tracking for first-party cookies. See "Specifying Session Tracking for First-Party Cookies" on page 65

Implementing the JavaScript Tag

Implement the Webtrends JavaScript tag on all the pages that you want to track. You can place the JavaScript tag anywhere between the <body> and </body> tags on a web page. Placing the tag at the top of the page directly after the <body> tag allows the tag to execute even if the page does not fully load. However, we recommend you place it at the bottom of the page just before the </body> tag to ensure the JavaScript tag is only activated after the page fully loads and all the information that the tag needs is available.

In addition to placing the tag directly in your web pages, there are other methods for tagging your pages which can make it easier to tag many pages quickly. Alternatively, you can place the tag in a client-side include file or place the tag in a footer template. For more information on these options and tagging best practices, see "Client-Side JavaScript Integration" in the *Webtrends Analytics Implementation and Maintenance Guide*.

Specifying the First-Party Cookie Data Source

In this step, you edit a profile and specify the first-party cookie data source for that profile.

Note

If your web site has multiple domains and you want to tracking visitors across them, you must create a separate data source for each domain. For more information, see "Configuring Domains" on page 65 and "Tracking Visitors Across Domains" on page 66.

To specify the first-party cookie data source:

1. In the left pane, click **Administration > Web Analysis** > **Reports & Profiles**.
2. Mouse over a profile and click **Edit** on the Action menu.
3. Click **Analysis > Data Sources**.
4. Click **New**. If you use Webtrends On Demand, you cannot specify a new data source. You simply need to specify the first-party cookie data source for this profile. Click **Save**, and go to "Specifying Session Tracking for First-Party Cookies" on page 65.
5. In the Data Sources dialog specify the name of the server and select the data source.
6. Click **Save**, and your new data source for first-party cookies appears in the list of data sources for that profile.

Specifying Session Tracking for First-Party Cookies

Session Tracking definitions determine how Webtrends identifies visits and counts unique visitors.

To specify session tracking:

1. In the left pane, click **Administration > Web Analysis** > **Reports & Profiles**.
2. Mouse over a profile and click **Edit** on the Action menu.
1. Click **Analysis** > **Session Tracking**.
2. Clear the **Always Use Default Definition** check box.
3. Click **Track User Sessions Using First Party Cookie**.
4. Click **Save**.

Configuring Domains

By default, the domain for the first-party cookie is populated with the actual domain that served the page. If you need to track visitors across different domains, make sure that you configure your JavaScript tag to set the domain that should be associated with your first-party cookie.

Consider the following domains:

- www.newstuff.webtrends.com
- www.standardstuff.webtrends.com
- www.ultra.cool.things.webtrends.com

All of these domains are subdomains of www.webtrends.com, which is a root domain. Therefore, in the Webtrends On Demand user interface, you would specify `.webtrends.com` (note the leading period) to track cookies across these domains.

Also, note that another domain such as `www.Webtrends.store.com` can be a separate root domain that belongs to the same Webtrends account. Using Webtrends cross-domain tracking, the same visitor ID can be moved from one domain to another. This is because Webtrends On Demand can recognize that all four domains are members of the same account (Webtrends).

You can specify the domain for your cookie when you add or edit a data source.

To specify your domain:

1. In the left pane, click **Administration > Application Settings > Data Sources**.
2. Mouse over a data source and click Edit on the Action menu.
3. Click **SmartSource Data Collector**.
4. Click **Tracking**.
5. Select the **Set the First-Party Cookie domain** check box.
6. Type the name of the domain you want to use. Be sure to precede the domain name with a period. Doing so insures that all sub-domains are rolled up into the domain. If you do not add the period, the cookie is set to the actual domain which serves the page. For example, type `.Webtrends.com`

Tracking Visitors Across Domains

If you use Webtrends Analytics On Demand or Webtrends Analytics software with SmartSource Data Collector and you have multiple domains, your visitors will have a different first-party cookie set on each domain as well as a third-party cookie for your account. They will be reported as unique visitors to each domain when you use first-party cookie session tracking methods. However, Webtrends can track your first-party cookies across domains by using the third-party cookie that identifies your domain.

You can create a separate profile to track your visitors across your domains using the Account Rollup data source, keeping in mind that this data source uses the Webtrends third-party cookie for tracking visitors. The visit and visitor counts will be different when using this Account Rollup data source, compared to your more reliable first-party cookie profiles. However, it can provide you with meaningful insight into your account traffic if needed.

Note

Cross-domain tracking applies only to SDC data files. You cannot analyze both web server data and SDC data file and then perform cross-domain cookie tracking using the SDC account rollup data source.

As a best practice, Webtrends recommends that you use first-party cookies to identify enterprise-wide, cross-domain behavior and trends. This method leverages a Webtrends third-party cookie to establish the first-party cookie, which tracks visitors across the specified domains in your data sources. If the visitor rejects third-party cookies, the first-party cookies continue to identify the visitor; however, that particular visitor appears as a different visitor for each domain.

To track visitors across domains in Webtrends Analytics software:

1. Create a separate data source for each domain. For more information, see “Configuring Domains” on page 65.
2. In the left pane, click **Administration > Web Analysis > Reports & Profiles**.
3. Edit a profile.
4. Select **Analysis > Session Tracking**.
5. Select **Track User Sessions Using First-Party Cookie (Account Rollup)**. You might need to clear the **Always Use Default Definition** check box.
6. Click **Save**.

Note

You can make further modifications to your profile setting if necessary by editing the profile after you have saved it.

To track visitors across domains in Webtrends Analytics On Demand:

1. Create a separate data source for each domain. For more information, see “Configuring Domains” on page 65.
2. Create your tag using the Tag Builder at tagbuilder.webtrends.com.

3. Select **Web Analysis > Data Sources** and then select all of the data sources you want to analyze for this profile. Be sure that for each data source checked here, you generate the tag using **Tag Builder** and that you update the tag on every page of the domains affected. Missed pages will not be counted. The tag generated on the Data Sources tab does not support cross-domain tracking for Visitor Data Mart profiles.
4. Select **Web Analysis > Reports & Profiles**.
5. Edit the specified profile.
6. Select **Analysis > Session Tracking**.
7. Select session tracking options:
   - For Visitor Data Mart profiles, select the option that matches your selection under Visitor Data Mart Specific Visitor Tracking on the Advanced Settings tab in **Tag Builder**. The default selection in Tag Builder and in Session Tracking is **Use standard Webtrends Visitor ID (WT.vt_sid).**
   - For Webtrends Analytics profiles, select **Track User Sessions Using First-Party Cookie (Account Rollup)**.
8. Click **Next**.
9. Click **Save**.
10. The reports for this profile show aggregated data for the domains that belong to that account.

Note
Webtrends On Demand Business Edition does not support cross-domain tracking.

Converting Third-Party Cookies to First-Party Cookies

If you have an existing Webtrends On Demand account that uses third-party cookies, you can add the first-party cookie tracking to your JavaScript tag. After the tag is implemented on your site, visitors immediately begin to receive the first-party cookies. Webtrends On Demand and SmartSource Data Collector (SDC) have built-in logic to stitch visitor records together to ensure a smooth transition from the third-party cookie methodology to the new first-party cookie methodology.

To convert third-party cookies to first-party cookies:

1. Modify the JavaScript tag as described in "Configuring Webtrends for First-Party Cookie Tracking" on page 63.
2. Update all your web pages to use the new JavaScript tag.
3. Wait 24 hours until analysis of the visitors who were tracked using the third-party cookie session tracking has completed.
4. Change the profile so it uses the first-party cookie session tracking definition.

Customizing Tag-Generated First-Party Cookies

If you use Webtrends On Demand or Webtrends software with SDC, you can customize the persistence and expiration date of your first-party cookie.

Creating Session Cookies

If you want to generate session cookies rather than persistent cookies, you remove the expiration date parameter from the cookie. However, this is not recommended because Webtrends cannot use session cookies to accurately track unique visitors to your site. Also, visit counts are inaccurate if the visitor closes the browser, reopens it and immediately returns to your site. In this case, the visitor is identified as a new unique visitor, and the visit is considered a new visit to your site.

To create session-based first-party cookies:

1. In the left pane, click **Administration > Application Settings** > **Data Sources**.
2. Edit the data source.
3. Click **SmartSource Data Collector**.
4. Modify the following line from the JavaScript tag text box:

```
var expiry="; expires="+dExp.toGMTString();
```

so that it looks like this:

```
var expiry="";
```

5. Click **Download this tag** to save your new tag.
6. Click **Save**.
7. Implement the tag on your web site, replacing any existing tags, and redeploy the updated pages to your web site.

Since session cookies are only valid for the current visit, they cannot be used to accurately report on many aspects of visitor data, including unique visitors, campaign tracking, commerce tracking, search engine history, and other visitor history based analysis reports. Because the use of session cookies may alter the statistics in reports from what you are used to viewing, you should try session cookies on one profile as a test model and look at the numbers in the resulting report to see if that is what you were expecting. After you accept the results, you can apply session cookies to other profiles.

Configuring Cookie Expiration

With Webtrends v7.5 and higher, the first-party cookie set by the JavaScript tag is configured to expire in 10 years. You can change the expiration by modifying the time value parameter in the statement.

To configure the first-party cookie expiration:

1. In the left pane, click **Administration > Application Settings** > **Data Sources**.
2. Edit the data source.
3. Click **SmartSource Data Collector**.

4. Edit the following line from the JavaScript tag text box:

   ```
   var dExp=new Date(dCur.getTime()+315360000000);
   ```

5. `315360000000` represents the total number of milliseconds in 10 years. Change this value to the number of milliseconds from the current time until the time that you want the cookie to expire. For example, if you want the cookie to expire in 60 days, then change this value to 5183940000 = 60 (days) * 24 (hrs per day) * 60 (minutes per hour) * 60 (seconds per minute) * 1000 (milliseconds per second.).
6. Implement the modified tag, replacing the existing tags on your pages, and redeploy the updated pages to your web site.

Using First-Party Cookies Without SDC

If you use Webtrends Analytics software without SDC, you can use first-party cookies to identify visitors simply by creating a Session Tracking definition that specifies your cookie and applying it to your profiles.

To create a Session Tracking definition:

1. In the left pane, click **Administration** > **Web Analysis** > **Options** > **Session Tracking**.
2. Click **New**.
3. Type a name for the definition in the **Description** field.
4. Select **Use the following alternate method(s)**.
5. Select the **Cookie** check box.
6. In the Track Sessions by Cookie dialog, select **Use this cookie.**
7. Specify the name of the cookie that your web server uses to identify visitors. For example, `Webtrends_ID` is the name of the cookie that the Webtrends Cookie Plug-in uses by default.

To use cookie session tracking in profiles:

1. In the left pane, click **Administration** > **Web Analysis** > **Reports & Profiles**.
2. Add or edit a profile.
3. Click **Analysis** > **Session Tracking**.
4. Clear the **Always use default definition** check box.
5. Select the cookie tracking definition that you created.
6. Click **Save**.

Implementing the Opt-Out Cookie

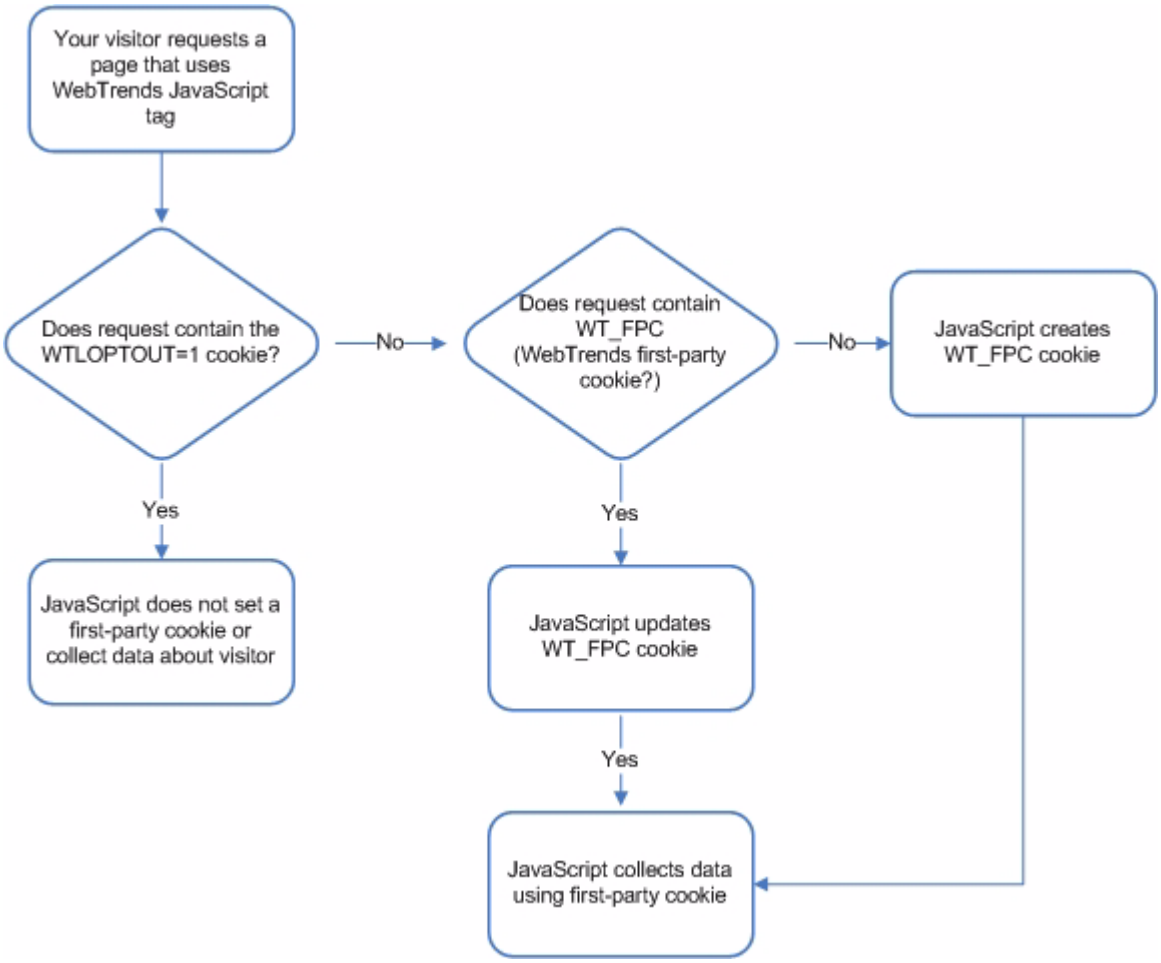
Because of misconceptions about the nature of cookies, some of your visitors might have concerns about how cookies that your web site generates might be used to track their Internet behavior. Using the Webtrends JavaScript, you can implement a method that allows visitors to your site to opt out of being tracked by your first-party cookie. The JavaScript supports *full opt-out*, in which the JavaScript does not collect data about the visit and does not set a cookie.

Implementing the full opt-out cookie involves the following high-level steps:

1. Write a policy about how your organization uses first-party cookies and post that policy on your site. For more information, see "Writing an Opt-Out Policy" on page 71.
2. Create a web page that allows your visitors to specify their tracking preference, and implement a method on your web site to set the full opt-out cookie, `WTLOPTOUT=1`. For more information, see "Creating an Opt-Out Mechanism" on page 71.
3. Deploy a Webtrends v8 or higher JavaScript tag on your site. For more information, see "Implementing the JavaScript Tag" on page 72.

How the Opt-Out Cookie Works

After you implement the opt-out cookie on your web site, the Webtrends JavaScript tag makes sure that no data is collected about visitors who have requested full opt-out. The following graphic shows the logic involved:



Writing an Opt-Out Policy

As a best practice, you should create a policy that describes how your organization uses first-party cookies and the differences between anonymous opt-out and full opt-out. Make it clear to visitors that accepting the opt-out cookie from your web server is the mechanism from preventing your first-party tracking cookie from being set and prevents data about their activity your on your site from being collected. Cookie-wary visitors should be aware that if they delete the opt-out cookie, your web server identifies them as a new visitor and sets a first-party cookie that can be used to collect data.

You can view the opt-out language that Webtrends Inc. uses at the following URL:

`http://ondemand.Webtrends.com/support/optout.asp`

Creating an Opt-Out Mechanism

Your web site developer will need to implement a way to set the opt-out cookie for visitors who specify that they do not want tracked on your site. The JavaScript tag looks for a cookie named `WTLOPTOUT` set to a value of `1`.

Because cookies are domain-specific, if you have multiple domains, you need to set the opt-out cookie for each domain. If you have multiple sub-domains, use the domain attribute in the cookie to specify the main domain. For example, `ondemand.Webtrends.com` is a sub-domain of `webtrends.com`. In order to set a cookie that is identified for both the domain and all sub-domains, the web site developer passes `domain=.webtrends.com` for the domain attribute as shown in the following example:

```
WTLOPTOUT=yes expires= Thu, 14 Jan 2016 18:40:50 UTC; path=/: domain=.Webtrends.com
```

The following example shows JavaScript that sets a full opt-out cookie:

```
<html>
<body>

<script type="text/javascript"><!--

// test for existence of WTLOPTOUT cookie
if (document.cookie.indexOf("WTLOPTOUT=")==-1){

// compute cookie expiration
var dCur=new Date();
var dExp=new Date(dCur.getTime()+315360000000);

// initialize cookie attributes
var expiry="; expires="+dExp.toGMTString();
var path="; path=/";
var domain="; domain=.Webtrends.corp";

// WTLOPTOUT cookie does not exist so set it
document.cookie="WTLOPTOUT=1"+expiry+path+domain;
}

//-->
</script>

</body>
</html>
```

Implementing the JavaScript Tag

If you used earlier versions of Webtrends Analytics software or Webtrends Analytics On Demand, you need to generate a new JavaScript tag for each data source and update the tag on your site. For more information, see “Implementing the JavaScript Tag On Your Web Pages” in Administration Help.

Disabling Cookies

You can configure Webtrends Analytics On Demand and SmartSource Data Collector to not set cookies if your organization’s policy does not allow visitor data to be tracked using cookies.

Disabling First-Party Cookies

If your organization decides not to set a first-party cookie, you can edit your SmartSource data sources and re-generate your JavaScript tag so that first-party cookie tracking is disabled.

To disable first-party cookie tracking:

1. In the left pane, click **Administration > Application Settings > Data Sources**.
2. Mouse over a SmartSource data source and click **Edit** on the Action menu.
3. Click **SmartSource Data Collector**.
4. Click **Tracking**.
5. Clear the **Enable First-Party Cookie Tracking** check box.
6. Click **Generate Tag**.
7. Implement the new JavaScript tag on your web site.

Disabling Third-Party Cookies

If you use first-party cookies, you may want to disable the third-party cookies that are set by Webtrends Analytics On Demand or SmartSource Data Collector. Although setting both first-party and third-party cookies may seem unnecessary, there are several reasons that you should consider using both. Webtrends uses third-party cookies for the following purposes:

- To set a new first-party cookie for returning visitors who previously were only identifiable by the third-party cookie. If you disable third-party cookie tracking, all visitors are considered new visitors until the first-party cookie is set. This only a consideration if you are upgrading from an earlier version.
- To track visitors across multiple domains.
- To identify visitors for session tracking when third-party cookie is available, but a first-party is not available.

Disabling Third-Party Cookies for Webtrends On Demand

You can prevent Webtrends On Demand from setting third-party cookies for all hits by editing your data source and adding the following line to the JavaScript tag:

```
DCS.dcscfg=1;
```

Disabling Third-Party Cookies for SmartSource Data Collector

You can prevent SmartSource Data Collector (SDC) from setting third-party cookies.

- To prevent SDC from setting a third-party under any circumstance, edit the `dcs.cfg` file and set the `enabled` setting in the `[cookieserver]` section to `false`.
- To disable third-party cookies for all hits using the JavaScript tag, edit the `dcs.cfg` file and add or edit the following line: cfgbyhit=true. Also, edit your data source and add the following line to the JavaScript tag: `DCS.dcscfg=1;`

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate branding updates.
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Chapter 9

Webtrends Query Parameter Reference

Web analytics is about making business sense out of web visitor behavior – the same common-sense analysis that business professionals have been applying to traditional offline business for decades. Gaining the appropriate insight to enable your organization to make smarter business decisions means understanding your site's business objectives and determining the appropriate web metrics to provide that information.

Today's business environment typically involves working at remote and off-site locations. Consequently, the people responsible for the web site and those analyzing visitor activity on the web site to measure web metrics often do not work side-by-side. That is, they are frequently in different groups and locations. This situation necessitates coordination between developing and delivering content, and reporting and analyzing the activity results.

To facilitate interaction between departments, locations and individuals, Webtrends provides a feature designed to place the power of analysis into the hands of those individuals who are most interested in it. This feature, Webtrends query parameters, uses typical web site instrumentation to facilitate the analytics end-users want to see.

If you are reading this chapter in the online Help, you can quickly find any parameters of interest by searching the Help. If you want to read the entire chapter, use the Help's Table of Contents to see all the topics in this chapter.

How Webtrends Query Parameters Work

By implementing Webtrends query parameters when designing your web site, reporting becomes a part of the web site design process, eliminating the considerable coordination required and reducing the misunderstandings that can happen when the objectives of many groups are at play. Webtrends query parameters are passed in URLs through JavaScript, captured in SmartSource Data Collector log files, and ultimately used by Webtrends Analytics software or Webtrends Analytics On Demand. Webtrends Analytics uses these parameters to analyze your web activity and to produce reports.

Using Webtrends Query Parameters

If one of your goals is to limit the amount of manual intervention that is required to produce valuable reports, you can implement Webtrends query parameters in your web pages to automate the configuration of many features in Webtrends Analytics, making the Webtrends solutions more insightful with reduced effort.

For example, you can use Webtrends query parameters in HTML META tags to automatically configure page titles, product names and product information, or campaign names. You could also implement a custom “Campaign Submission” page in a company intranet that sends the Webtrends query parameter to automatically configure Webtrends Analytics to report on new campaigns, allowing you to spend less time on system configuration.

Products Using Webtrends Query Parameters

The following Webtrends products take advantage of Webtrends query parameters:

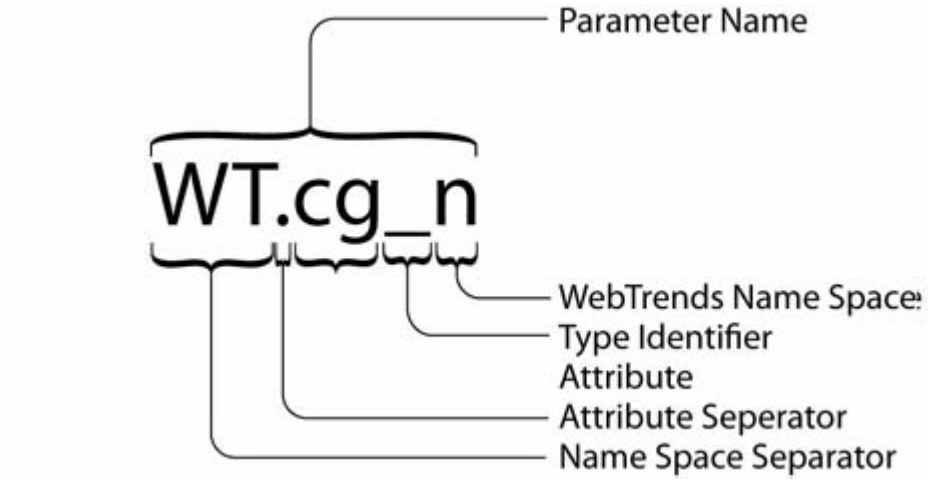
- SmartSource Data Collector
- Webtrends Analytics On Demand
- Webtrends Analytics software
- Webtrends Visitor Data Mart
- Webtrends Visitor Intelligence
- Webtrends Score

Query Parameter Syntax

Webtrends query parameters follow specific syntax that includes name, value, and format. The following sections describe each of these syntax elements

Name Syntax

The name of each Webtrends query parameter uses the following syntax:



Although Webtrends Analytics handles query parameter names without considering case, JavaScript object names are case-sensitive. Therefore, if you want JavaScript tagging to capture events, you must adhere to specified naming conventions. Webtrends reserved query parameters require an upper case name space, either `WT` or `DCS`, and the type identifier in lower case.

In addition to the query parameters covered in this chapter, the following name spaces are reserved for Webtrends:

WT.i_
Used for product integrations with Webtrends partners.

WT.z_
Used by Webtrends Professional Services for customer integrations.

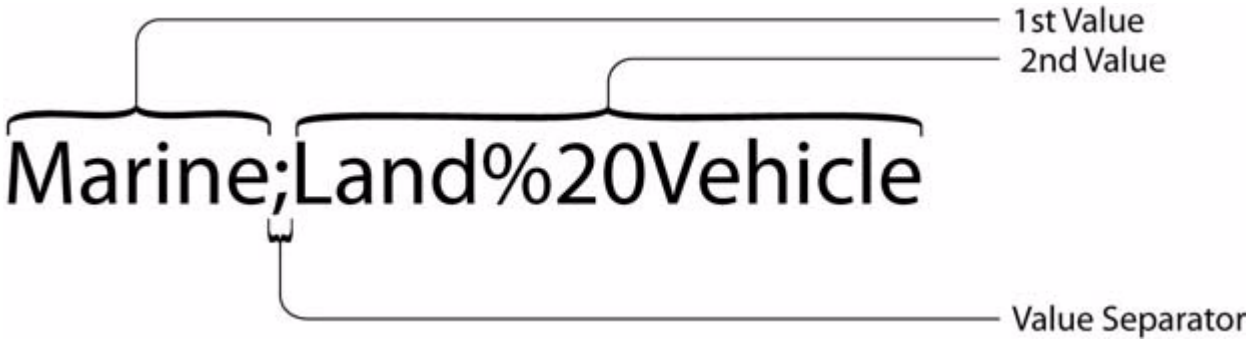
As a best practice, use the following syntax to create custom query parameters:

`DCSext.w_`*`custom_identifier`*

For example, you could create a parameter called `DCSext.w_articleid` to track an article ID.

Value Syntax

The values associated with Webtrends query parameters use the following syntax.



Each Webtrends query parameter name may have one or more values. Some parameters can be specified in pairs or in groups of related parameters. When related parameters have multiple values, these values may be correlated and their position becomes important as shown in the following example:

```
WT.si_n=name1;name2&WT.si_x=position1;position2.
```

In the previous example, if correlation is specified in the report, `name1` is associated with `position1` and `name2` with `position2`.

If there are multiple values, they are typically separated using a semicolon (;).You can use other separators, but you must specify the separator in the dimension or measure setting that is based on the parameter.

Note
Not all Webtrends query parameters support multiple values. For example, HTML Title Page allows only one value.

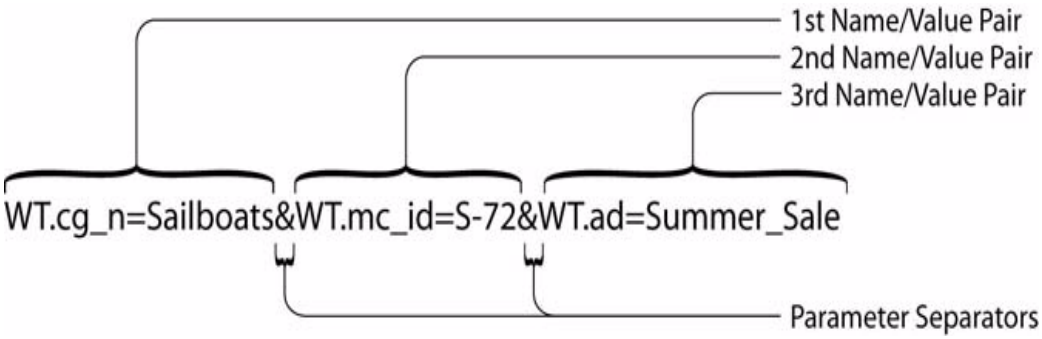
If the parameter value contains a semicolon, it must be hex-encoded (“%3B”) to differentiate it from the separator.

Numerical Value Format

Unless stated otherwise, numerical values must be specified using the simple US format using the period as the decimal separator with up to 2 decimals and without thousand separators. For example, 12345.67.

Complete Syntax

The Webtrends query parameters are represented as name/value pairs. The name/value pairs adhere to the following syntax:



The Webtrends query parameters are logged in SmartSource Data Collector log files. Each name/value pair is separated by an ampersand (&).

The Webtrends query parameters must co-exist with other web site well-known parameters. This means that Webtrends query parameters can be mixed with standard well-known parameters. The Webtrends query parameters can be separated from the other well-known parameters using the ampersand delimiter. While a Webtrends namespace is part of each query parameter, it is conceivable that parameter collisions may occur. If a query parameter string contains duplicate key values, the first instance is used and the others are discarded.

Syntax Examples

Single Parameter With a Single Value

The following example shows a page associated with the “Finance Offer” advertising view:

```
WT.ad=Finance%20Offer
```

Single Parameter With Multiple Values

The following example shows a page associated with both the “Finance Offer” and “FishFinder Offer” advertising views:

```
WT.ad=Finance%20Offer;FishFinder%20Offer
```

Related Parameters With a Single Value

The following example shows a page associated with the Campaign “New Product” and the Campaign Event Type “click.”

```
WT.mc_id=New%20Product&WT.mc_ev=click
```

Related Parameters With Multiple Values

The following example shows a page associated with the Campaign "Get Results" and "New Product" with the Campaign Event Type "Click." Note that because the second parameter uses a single value, if correlation is used, the "Click" value of the second parameter is associated with both values of the first parameter.

```
WT.mc_id=Get%20Results;New%20Product&WT.mc_ev=Click
```

Types of Query Parameters

Webtrends query parameters consist of the following groups:

Auto-configuration parameters
Recognized by Webtrends and used to auto-configure certain features. For more information, see "Auto-Configuration Parameters" on page 80.

Custom report parameters
Associated with preconfigured custom reports. For more information, see "Custom Report Parameters" on page 86.

SmartView parameters
Used by Webtrends SmartView. For more information, see "SmartView Parameters" on page 97.

Stored visitor parameters
Used for identifying visitors when the Visitor History database is exported. For more information, see "Stored Visitor Parameter" on page 98

Visitor History parameters
Related to visitor properties added by Webtrends during the analysis process. For more information, see "Visitor History Parameters" on page 99.

SDC-generated visitor parameters
Generated by SmartSource Data Collector (SDC). For more information, see "SDC-Generated Visitor Parameters" on page 112.

SDC-parameter override parameters
Override SDC parameters on the client side. For more information, see "SDC-Parameter Override Parameters" on page 119.

Conversion plug-in parameters
Used by the encoding conversion plug-in. For more information, see "Conversion Plug-In Parameters" on page 121.

Content reports parameters
Used to include web 2.0 content information in reports. For more information, see "Content Parameters" on page 95.

Auto-Configuration Parameters

Webtrends uses certain query parameters to automatically configure the appropriate advanced features and create related reports. The following types of auto-configuration parameters are available:

- Content Group (see "Content Group Parameters" on page 81)

- Marketing Campaign (see "Marketing Campaign Parameter" on page 81)
- Advertising View (see "Advertising View Parameter" on page 81)
- Advertising Click (see "Advertising Click Parameter" on page 82)
- Server (see "Server Parameter" on page 82)
- Scenario Analysis (see "Scenario Analysis Parameters" on page 82)
- Title (see "Title Parameter" on page 86)
- Split (see "Split Parameter" on page 86)

Content Group Parameters

A Content Group definition uses the following parameters:

- `WT.cg_n`
- `WT.cg_s`

You can specify multiple content groups per page. The Sub-Content Group parameter is optional. If none of the Content Groups on a particular page contain Sub-Content Groups, Webtrends Analytics may ignore the `WT.cg_s` parameter. If the `WT.cg_s` parameter is included, each Content Group value must have an associated Sub-Content Group value. The Sub-Content Group value may be empty.

Visitor Data Mart also uses this parameter. For more information about the complete set of query parameters that Visitor Data Mart uses, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in *Webtrends Visitor Data Mart User's Guide.*

`WT.cg_n`

`WT.cg_n=`*`Name[;…]`*

This parameter identifies the name of a Content Group. The maximum length is for each *`Name`* is 64 bytes.

`WT.cg_s`

`WT.cg_s=`*`SubName[;…]`*

This parameter identifies the name of a Sub-Content Group. This parameter is optional. The maximum length is for each *`SubName`* is 64 bytes.

Marketing Campaign Parameter

With Webtrends v7.0 and higher, `WT.mc_id` replaces `WT.mc_n` and `WT.mc_t`. For more information, see "Campaign Parameter" on page 93.

Advertising View Parameter

The Advertising View definition uses the `WT.ad` parameter, which supports multiple Advertising Views per page.

`WT.ad`

`WT.ad=`*`Name[;…]`*

The name of the advertisement viewed on a particular web page. The maximum length for each *`Name`* is 64 bytes.

Advertising Click Parameter

The Advertising Click definition uses the `WT.ac` parameter, which supports a single Advertising Click per page. If a page contains multiple ads, you can design the page to respond to clicks so that each click generates a hit to SDC.

`WT.ac`

`WT.ac=`*`Name`*

The name of the advertisement clicked to reach a particular web page. To capture this information, the Advertising Click must contain an external redirect back to the client. The redirect needs to include the necessary code to generate a hit to SDC or Webtrends On Demand. The maximum length for each *`Name`* is 64 bytes.

Server Parameter

The Server definition uses the `WT.sv` parameter, which supports a single server per page.

`WT.sv`

`WT.sv=`*`Name`*

This parameter identifies the name of the web server that served the web content. This is used for server cluster load balanced reports. The maximum length for *`Name`* is 256 bytes.

Note

The `WT.sv` parameter requires that the profile be set up as clustered with at least one server. This is the server that is used if an incoming hit does not explicitly have a server defined.

Scenario Analysis Parameters

Scenario Analysis parameters specify well-known paths or processes in your web site and are typically used to measure conversion and abandonment.

In order to report on Scenario Analysis in Webtrends Analytics, you must also specify your Scenario Analysis definitions in Webtrends Administration.

Visitor Data Mart also recognizes these parameters. For more information, see “Configuring Your Web Site to Collect Visitor Data Mart Data” in the *Visitor Data Mart User’s Guide.*

Scenario Analysis definitions use the following attributes:

`WT.si_n` – Name of the Scenario Analysis

`WT.si_p` – Identifies the step by name

`WT.si_x` – Identifies the step by position

`WT.si_cs` – Identifies the step in which conversion occurs for Visitor Data Mart profiles.

You can configure multiple Scenario Analyses per page. However, you cannot count the same page as two different steps in the same scenario.

Note

Because Scenario Analysis dimensions and measures are visit-based, they should only be used with visit-based dimensions and measures. If you create a custom report that uses Scenario Analysis dimensions with a hit-based dimension such as Page Views, the reported page view count will include all the page views of every visitor who saw one of the steps of the scenario, which inflates the actual page view count.
For example, if `WT.pn_sku`, which is hit-based, is specified on the same page as the Product View scenario step, and you create a custom report using Product SKU as the dimension and four scenario steps as measures, the Product SKU will show a visit for each step of the Scenario, even though it was only specified on one of the steps

WT.si_n

`WT.si_n=`*`Name[;…]`*

Identifies the name of the Scenario Analysis. The maximum length for each *`Name`* is 64 bytes. When configuring a new Scenario Analysis definition using Webtrends Administration, make sure that the Scenario Analysis name matches this parameter value.

When specifying multiple Scenario Analysis names, you must also specify multiple step names or step numbers. Each step name or step number is associated with the scenario name having the same position in the list.

WT.si_p

`WT.si_p=`*`StepName[;…]`*

Note

- Webtrends Analytics supports this parameter for Analytics Reports. Use `WT.si_x` to identify steps for Visitor Data Mart.
- Do not use `WT.si_p` if you specify the step by position using the `WT.si_x` parameter.

Identifies the step by name. When configuring a new Scenario Analysis step using Webtrends Administration, the step name needs to match this parameter value.

WT.si_x

`WT.si_x=`*`StepPosition[;…]`*

Identifies the step by numeric position. The value for `WT.si_x` must be an integer 1 or higher, and values need to be sequential.

Note

Do not use if specifying the step by name using the `WT.si_p` parameter.
Also, for Visitor Data Mart only, to translate the position numbers displayed in Webtrends Explore into something more meaningful, create a Scenario Analysis Definition in Webtrends Administration. Ensure that you configure a step name for each step.

WT.si_cs

`WT.si_cs=`*`1/0[;...]`*

Identifies whether the page is the step in which conversion occurs when the value is 1. Most likely, this page is the last step in your Scenario.

In addition to this parameter, the web server must also pass the step query parameter for this page. Webtrends Analytics recognizes either `WT.si_p` or `WT.si_x` for steps. Visitor Data Mart recognizes `WT.si_x` for steps.

Multiple values are supported. If multiple values are passed for this parameter and other Scenario event parameters, Visitor Data Mart correlates the values.

Visitor Data Mart recognizes this parameter. However, this parameter is currently not included in any Webtrends Analytics preconfigured custom reports.

Scenario Analysis Parameters for Shopping Cart Analysis

Use Scenario Analysis parameters to track shopping cart activity. Webtrends Analytics supports these parameters for the purchase conversion funnel report. Visitor Data Mart supports these parameters for multi-dimensional reporting through Webtrends Visitor Intelligence and for visitor segmentation in Webtrends Explore.

Use the following parameter names and values to tag your shopping cart pages. If you use Webtrends Visitor Data Mart or think that you might in the future, use the Step by Number method. If you use Webtrends Analytics, you can use either the Step by Name method or Step by Number method.

Step by Name Method (WT.si_p)	Step by Number Method (WT.si_x)	Query Parameter Purpose and Use
`WT.si_n=ShoppingCart`	`WT.si_n=ShoppingCart`	Identifies the shopping cart scenario. Webtrends Analytics uses the ShoppingCart value to create a purchase conversion funnel report. Visitor Data Mart uses the ShoppingCart value to provide top-level Scenario data.
`WT.si_p=CartView`	`WT.si_x=1`	Identifies the product view step
`WT.si_p=CartAdd`	`WT.si_x=2`	Identifies the cart start step
`WT.si_p=CartCheckout`	`WT.si_x=3`	Identifies the cart check out step
`WT.si_p=CartComplete`	`WT.si_x=4`	Identifies the cart complete step
`WT.si_cs=1`	`WT.si_cs=1`	Identifies the step in which conversion occurs

To collect shopping cart activity for reporting, you need to set up your purchases pages with shopping cart query parameters and create a Scenario Analysis definition in Webtrends Administration. For information about Scenario Analysis definitions, see Administration Help.

To implement shopping cart scenario parameters using WT.si_p:

1. Configure the product detail pages where visitors can click a button to buy a product. These pages differ from the shopping cart basket page that contains all items in the cart.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_p=CartView` to identify tagged pages as the first step.
2. Configure the pages where your visitors add or remove items from the shopping cart.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_p=CartAdd` to identify tagged pages as the second step in the scenario.
3. Configure the pages where your visitors start the checkout process.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use`WT.si_p=CartCheckout` to identify tagged pages as the third step in the scenario.
4. Configure the page where your visitors have successfully completed the purchase:
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_p=CartComplete` to identify tagged pages as the fourth step in the scenario.

To implement shopping cart scenario parameters using WT.si_x:

1. Configure the product detail pages where visitors can click a button to buy a product. These pages differ from the shopping cart basket page that contains all items in the cart.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_x=1`. to identify tagged pages with the first step in the scenario.
2. Configure the pages where your visitors add or remove items from the shopping cart.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_x=2` to identify tagged pages with the second step in the scenario.
3. Configure the pages where your visitors start the checkout process.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_x=3` to identify tagged pages with the third step in the scenario.
4. Configure the pages where your visitors have successfully completed the purchase.
   a. Use `WT.si_n=ShoppingCart` to identify tagged pages with your shopping cart scenario.
   b. Use `WT.si_x=4` to identify tagged pages with the fourth step in the scenario.

c. *If you want to track the purchase complete page as the conversion step for use in Visitor Data Mart*, use `WT.si_cs=1`.

Note

In order to provide useful names for Webtrends Explore and Webtrends Visitor Intelligence users, you also need to create a Scenario Analysis definition through Webtrends Administration. Ensure that you configure a step name for each step.

Title Parameter

The Title parameter, `WT.ti`, supports a single page title per page.

`WT.ti`

`WT.ti=`*`Title`*

The HTML title of the associated web content. If this parameter is found in parameter list, the value is used in the reports. When present, no other page title lookups are performed. The maximum length for *`Title`* is 1024 bytes.

`WT.ti` always overrides other methods of page title lookup. In a case where two different pages (both having `WT.ti`) end up being rebuilt to the same URL (via URL Rebuilding), the last page seen wins.

Split Parameter

The Split parameter, `WT.sp`, is the default parameter Webtrends uses to split log files for Parent Child profiles. The parameter supports multiple values per page in order for any given page to be part of multiple Child profiles.

`WT.sp`

`WT.sp=`*`ProfileName[;…]`*

The Split parameter is used by the Parent Child profile generator to build child profiles. As a best practice, keep the length of this parameter as short as possible. The format of this value is an alphanumeric string (for example, `UTF-8`).

Custom Report Parameters

This section discusses parameters that are included in a series of preconfigured custom reports. These query parameters are grouped into categories as follows:

- Search Engine – see “Search Engine Type Parameter” on page 87.
- Web Client – see“Web Client Parameters” on page 87.
- Transaction – see “Transaction Parameters” on page 91.
- Invoice – see “Invoice Parameters” on page 93.
- Campaign – see “Campaign Parameter” on page 93.
- Campaign Event – see “Campaign Event Parameter” on page 94.
- Content – see “Campaign Event Parameter” on page 94.

Search Engine Type Parameter

WT.srch

WT.srch=*SearchEngineType*

The Search Engine Type parameter in the query string of the URL from a referring search engine identifies the link as one for a paid search engine phrase.

Webtrends Analytics uses this parameter for Custom Reporting and Visitor History to differentiate a paid search engine from a reference to an organic search engine. Only `WT.srch=1` has meaning. No other values are defined or recognized.

Note

Search engine results using `WT.srch=1` are limited to only those search engines that are listed on the Webtrends search engine availability list. Using `WT.srch=1` will not produce search engine campaign results for search engines that are not recognized by Webtrends.

Visitor Data Mart also uses this parameter. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in *Visitor Data Mart User's Guide*.

Web Client Parameters

This section describes parameters that are associated with web client (browser) properties. These parameters are typically used for creating custom dimensions that you include in custom reports.

WT.tz

WT.tz=*Timezone*

Indicates the web client's time zone and is the offset of the web client from UTC. The value is expressed in hours.

Positive values are given without a sign (`WT.tz=2`), but negative values require a minus sign (`WT.tz=-12`). Values can range from `-12` to `+14`.

Example:

```
var dCurrent = new Date();
var tzQueryParam = "&WT.tz=" + escape(dCurrent.getTimezoneOffset());
```

To represent Pacific Standard Time, use the following parameter:

```
WT.tz=-8
```

WT.bh

WT.bh=*BrowsingHour*

Indicates the web client's local time of day on a 24-hour clock. Values can range from 0 to 23.

WT.ul

WT.ul=*UserLanguage*

Indicates the web client's user language.

Example:

```
var ulQueryParam = "&WT.ul=" + navigator.appName == "Netscape" ?
escape(navigator.language) : escape(navigator.userLanguage);
```

WT.cd

WT.cd=*ColorDepth*

Indicates the web client's screen color depth. It is represented as the number of color bits to which the web client computer's video display control is set.

Example:

```
var cdQueryParam = "&WT.cd=" + escape(screen.colorDepth);
```

WT.sr

`WT.sr=`*`ScreenResolution`*

Indicates the web client's screen resolution. It is expressed as the gross width and height of the web client's monitor. The format of the value is *`widthXheight`* (for example, `800X600`).

Example:

```
var srQueryParam = "&WT.sr=" + escape(screen.width) + "x" + escape(screen.height);
```

WT.jo

`WT.jo=`*`IsJavaEnabled`*

Indicates if the web client has enabled Java. Valid values are Yes and No.

Example:

```
var joQueryParam = "&WT.jo=" + navigator.javaEnabled() ? "Yes" : "No";
```

WT.js

`WT.js=`*`IsJavaScriptEnabled`*

Indicates whether the web client supports and/or has enabled JavaScript. Valid values are `Yes` and `No`.

For example:

```
<SCRIPT LANGUAGE="JavaScript">
...CODE ABOVE WHERE YOU WANT THIS MODIFICATION TO GO...
var jsQueryParam = "&WT.js=Yes";
...CODE BELOW WHERE YOU WANT THIS MODIFICATION TO GO...
</SCRIPT>

<NOSCRIPT>
<IMG BORDER="0" NAME="DCSIMG" WIDTH="1" HEIGHT="1" SRC="http://localhost/
njs.gif?dcsuri=/nojavascript&WT.js=no">
</NOSCRIPT>
```

Note

You can use Webtrends to set up a custom reports (augmenting the Pages reports) that keys on **`WT.js`** and provides a targeted count of no JavaScript hits.

WT.jv

`WT.jv=`*`JavaScriptVersion`*

Indicates the version of JavaScript supported by the web client. If JavaScript is not enabled, this parameter should not be included.

Example:

```
var jvQueryParam = "";
<SCRIPT Language"Javascript1.0">
jvQueryParam = "&WT.jv=1.0"
</SCRIPT>

<SCRIPT Language"Javascript1.1">
jvQueryParam = "&WT.jv=1.1"
</SCRIPT>
```

```
<SCRIPT Language"Javascript1.2">
jvQueryParam = "&WT.jv=1.2"
</SCRIPT>
```

WT.ct

WT.ct=*connectiontype*

Identifies the visitor's connection type. You can use this parameter in custom reports to determine whether visitors can download media on your site that requires high-bandwidth connections.

The JavaScript tag generates this value using data from Microsoft Internet Explorer 5 or higher. Valid values are `lan`, `modem`, and `offline`. For all other browsers, the JavaScript tag generates a value of `unknown`.

WT.hp

WT.hp=*isHomePage*

Indicates whether your visitors have configured a URL as their home page. It is only available for visitors using Microsoft Internet Explorer 5 or higher. Valid values are 1 and 0.

You can use this parameter to filter a report based on URLs or pages, and report on pages that are used as home pages by visitors.

WT.bs

WT.bs=*browserSize*

Identifies the actual size of the visitor's browser window. It is expressed as the width and height of the web client window in pixels. The format of the value is widthxheight (for example, `924x212`).

WT.fi

WT.fi=*isFlashInstalled*

Indicates whether your visitors have installed the Macromedia Flash browser plug-in. Valid values are `Yes` and `No`.

For more information see, the `WT.fv` parameter.

WT.fv

WT.fv=*flashVersion*

Indicates the version of the Macromedia Flash browser plug-in if installed. Requires the presence of `WT.fi=Yes`.

web. For example, `WT.fv=7.0`.

WT.le

WT.le=*languageEncoding*

Specifies the character set used by the web client to render the current document. This parameter can be used to troubleshoot internationalization issues. The format of this value is an alphanumeric string (for example, `UTF-8`). Go to the following site for a list of valid values:

```
http://www.iana.org/assignments/character-sets
```

WT.mle

WT.mle=*metaLanguageEncoding*

Specifies the character encoding set by the web client to render the current document. If the page includes the `http-equiv="Content-Type"` META tag, the Webtrends JavaScript tag generates this parameter and passes it as an alphanumeric string (for example, `UTF-8`). The parameter is used for troubleshooting internationalization issues.

For example, the JavaScript tag generates a value of `WT.mle=UTF-8` for the following tag:

```
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
```

Note

In order for this parameter to be generated, you must set the `gI18n` variable in the JavaScript tag to `true`. For more information, see "Customizing the JavaScript Tag" in the *Webtrends Analytics Software Implementation and Maintenance Guide*.

Go to the following site for a list of valid values:
`http://www.iana.org/assignments/character-sets`

WT.em

`WT.em=esc/uri`

Specifies the encoding method supported by the web client. This parameter can be used to troubleshoot internationalization issues. Valid values are `esc` for the JavaScript `escape()` function and `uri` for the JavaScript `encodeURIComponent()` function.

WT.slv

`WT.slv=Silverlight_version`

Specifies the version of the Microsoft Silverlight plug-in installed on the visitor's web client. If Silverlight is not enabled, this parameter is not provided.

Products Parameters

You can use Products parameters to report on transaction activity for products on your site. Several preconfigured custom reports use these query parameters.

These parameters support multiple values except where noted. When multiple values are passed, the order of the values is important because they correlate to other parameters. For example, multiple product values for the `WT.pn_sku` parameter correlate to the number of units passed in the `WT.tx_u` parameter.

Note

You cannot use these query parameters as dimensions and measures in Scenario Analysis funnels. Products parameters are hit-based while scenario parameters are visit-based and should not be used together in reports.

WT.pn_sku

`WT.pn_sku=productSKU[;…]`

Identifies the SKU (a unique numeric identifier) of the product. Visitor Data Mart also uses this parameter. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in *Visitor Data Mart User's Guide*.

Note

`WT.pn_sku` replaces the `WT.pn` and `WT.pc` parameters. `WT.pn` represented the name of the product. `WT.pc` represented the category of the product.

WT.pn_id

`WT.pn_id=productID[;…]`

Optional: Identifies the product identifier of a product. If possible, product IDs should be unique values to preserve lookup data integrity.The following parameters are automatically added by the product translation process when the product SKU is found in the product translation file. For more information about translation files, see “Using Lookup Tables for Analytics Reports” in the *Administration User’s Guide*.

Note

Product IDs typically map to multiple Product SKUs. For example, a sporting goods company might have an item with a specific ID and several SKUs corresponding to various colors.

WT.pn_fa

`WT.pn_fa=productFamily[;…]`

Identifies the family of the product.

WT.pn_gr

`WT.pn_gr=productGroup[;…]`

Identifies the group of the product.

WT.pn_sc

`WT.pn_sc=productSubCategory[;…]`

Identifies the sub-category of the product.

WT.pn_ma

`WT.pn_ma=productManufacturer[;…]`

Identifies the manufacturer of the product.

WT.pn_su

`WT.pn_su=productSupplier[;…]`

Identifies the supplier of the product.

Transaction Parameters

WT.tx_t

`WT.tx_t=Type[;…]`

In earlier versions of Webtrends, this parameter was used to identify transaction types. Because Webtrends Analytics does not use this query parameter in any preconfigured custom reports, it is no longer supported.

WT.tx_u

`WT.tx_u=Units[;…]`

Identifies the quantity in the transaction. Pass a positive integer for this value.

If an order contains multiple products, separate the numbers of units for each product using a comma or semi-colon (configurable) in the `WT.tx_u` variable.

When associating this measure with the product dimension in a Webtrends Analytics custom report, make sure you select the correlation option to ensure that the first number of units is associated with the first product, the second number of units with the second product, and so on.

Visitor Data Mart also recognizes this parameter. For more information, see “Configuring Your Web Site to Collect Visitor Data Mart Data” in the *Visitor Data Mart User’s Guide*.

WT.tx_s
`WT.tx_s=`*`Subtotal[;…]`*

Identifies the total cost for each product in the order.

The format of this field must match the currency format that Webtrends Analytics is configured to analyze. Do not include a currency symbol and be sure pass the value in *`dollars.cents`* format. For example, if you globally define your currency as US-Dollars in Webtrends Analytics, the format of this parameter set to a $4500 cost is: `WT.tx_s=4500.00`. Do not include a currency symbol or a comma in the value.

If an order contains multiple products, the totals for each product should be separated by a comma or semi-colon (configurable) in the `WT.tx_s` variable.

When associating this measure with the product dimension in a custom report, make sure you select the correlation option to ensure that the 1st amount is associated with the 1st product, the 2nd amount with the 2nd product, and so on.

Visitor Data Mart also recognizes this parameter. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*.

WT.tx_e
`WT.tx_e=`*`event`*

Identifies a product-related event. Webtrends Analytics uses this parameter as qualifier in preconfigured measure definitions in order to determine which product to count for a measure. Webtrends Visitor Data Mart uses this value to qualify certain preconfigured events. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*. You can use the parameter to report on specific activities by creating a custom report filter that you apply to a custom report.

Note

Even if there are multiple values specified in the WT.pn_sku (Products), WT.tx_u (Units) and WT.tx_s (Revenue) parameters, the WT.tx_e should contain a single value (the same event applies to the entire page view).

You can pass a custom value for this parameter, or if you want to track product purchases, product views, product cart additions, and product cart removals, pass one of the following values:

WT.tx_e=p
Required by Webtrends Analytics and Visitor Data Mart to identify a product purchase. Although Webtrends Analytics does not use `WT.tx_e=p` to determine whether a visitor is a buyer (`WT.vr_brws`), Visitor Data Mart does. Instead Webtrends Analytics uses `WT.tx_s` to determine whether the visitor is a buyer. In addition, you can use the Invoice query parameters to include invoice number, date, and time for the purchase. For more information, see "Invoice Parameters" on page 93.

WT.tx_e=v
Required by Webtrends Analytics and Visitor Data Mart to identify a product view.

WT.tx_e=a
Identifies a product cart addition.

WT.tx_e=r
Identifies a product cart removal.

WT.tx_cartid
`WT.tx_cartid=`*`CartIdentifier`*

Pass a unique value to identify a visitor's cart. Visitor Data Mart uses this parameter to identify events associated with a specific cart. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*.

Invoice Parameters

Use `WT.tx_i`, `WT.tx_id`, and `WT.tx_it` parameters together. You must enable Visitor History in order to use the Invoice parameters.

`WT.tx_i`

`WT.tx_i=`*`InvoiceNumber`*

Identifies the invoice number for the purchase. Webtrends Analytics uses data stored in Visitor History to make sure that a page view with an invoice number is a new purchase and not the result of a visitor refreshing the page after making a purchase. If Webtrends Analytics sees a page view with an invoice number, that page view is compared against the last three invoices for a visitor. If the `WT.tx_i` value does not match the last three invoices, Webtrends considers it a new purchase.

Visitor Data Mart also recognizes this parameter. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*.

Note

Webtrends Express Analysis cannot use the invoice data to identify duplicate purchase records because the Express Analysis Engine does not use visitor history data.

`WT.tx_id`

`WT.tx_id=`*`InvoiceDate`*

The format is `mm/dd/yy`. A 4-digit year is also allowed: `mm/dd/yyyy`.

Identifies the purchase invoice date, which is based on GMT. If the invoice date is three days older than the date of the visit, it is assumed that the hit is not an actual purchase but a view of a previously bookmarked or saved page. If the invoice date was less than three days than the date of the visit, the `WT.tx_i` parameter is used to determine if the hit is a valid purchase.

For example, a visitor makes a purchase. The purchase is accounted with an invoice date. The visitor saves a bookmark to the page. Five days later, the visitor goes to the bookmarked page. This causes another hit to be sent to SDC. However, the `WT.tx_id` parameter still contains the original purchase date. Webtrends analysis sees that the date of the hit is several days after the date found in the `WT.tx_id` parameter and determines that this is not an actual purchase.

`WT.tx_it`

`WT.tx_it=`*`InvoiceTime`*

Identifies time of the invoice. The format is `hh:mm:ss` where `hh` is in a 24-hour format (`0` = midnight, `23` = 11pm).

This parameter helps determine when an invoiced purchase was made. This value is used along with `WT.tx_id` and `WT.tx_i` to determine if the purchase was a valid purchase or if this was a user refreshing the web page after a purchase or returning to the page to check status.

Campaign Parameter

`WT.mc_id`

`WT.mc_id=`*`Campaign ID`*

Identifies a specific marketing campaign. Pass this query parameter to pages that you want to associate with a specific campaign. You can specify a numeric or string value.

Webtrends Analytics Considerations

If you plan to export Webtrends Analytics report data to a SmartReport, values that are either characters or a combination of numbers and characters work best with Microsoft Excel.

If you enable Visitor History in a profile, Webtrends Analytics reads this parameter and stores it in the Visitor History database. The most recent value of this parameter is made available by Visitor History on every hit as `WT.vr.rac`.

Webtrends Visitor Data Mart Considerations

Visitor Data Mart uses this parameter to identify Ad events. For more information about the parameter set used by Visitor Data Mart, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*.

Note

`NWT.mc_id` replaces the `WT.mc_n` and `WT.mc_t` auto-configuration parameters in earlier versions of Webtrends.

Campaign Event Parameter

`WT.mc_ev`

`WT.mc_ev=`*`EventType`*

This parameter identifies an ad event type.

Webtrends Analytics and Visitor Data Mart recognize the following values:

`WT.mc_ev=click` identifies an ad clickthrough event.

Webtrends Analytics Considerations

Webtrends Analytics does not currently provide preconfigured custom reports that use the **`click`** value. However, you can also specify a custom value to identify a custom event type, create a custom report filter based on this value, and apply it to a custom report.

Segment Parameter

`WT.seg_`*`X`*

`WT.seg_`*`X`*`=Segment`

Identifies a segment of interest. *X* can be 1, 2, 3, or 4. For example, `WT.seg_1=`*`Segment1`*. This parameter identifies values associated with this segment, and you can store these values in Visitor History. The Visitor History function recognizes these values and stores them in the Visitor History database. For more information, see "Visitor Segmentation Parameters" on page 112.

Page of Interest Parameter

`WT.pi`

`WT.pi=`*`Page identification`*

This parameter identifies a page on your site that is critical to evaluating performance. *Page identification* can be any string. When you enable **Page of Interest Unique Visitor Tracking** in the Visitor History dialog of a profile, Webtrends stores the values for the `WT.pi` parameter for each unique visitor in the Visitor History database. You can limit the amount of disk space used to store these values by keeping the strings as short as possible.

As a best practice, you should only identify key pages on your site with this parameter because Webtrends stores a maximum of 20 pages for each unique visitor for each profile.

Use this parameter to create a Page of Interest dimension that can be associated with a measure based on the `WT.vr.piv` Visitor History parameter. For more information, see `WT.vr.piv` on "Visitor Tracking Parameters" on page 110.

On-Site Search Parameters

On-site search parameters allow you to collect activity about your on-site search tool.

`WT.oss`

`WT.oss=`*Search phrase*

Identifies a word or a phrase that visitors submit for an on-site search.

`WT.oss_r`

`WT.oss_r=`*number of results*

Identifies whether or not an on-site search is successful. This parameter should be specified on the same hit as WT.oss and should be set to the number of results whenever the on-site search is successful, or to 0 when the search fails (no result).

Webtrends Analytics uses this parameter in preconfigured custom report filters. Webtrends Visitor Data Mart uses it provide data for the Number of Results attribute.

Registered Visitor Parameter

`WT.rv`

`WT.rv=1`

If you use Visitor Data Mart, your web server should pass this parameter with a value of 1 when a visitor has completed a registration process. For more information, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart Administration sGuide*.

Content Parameters

The parameters defined in this section can be used to populate custom reports that include web 2.0 content information.

`WT.rss_f`

`WT.rss_f=`*FeedName*

This parameter identifies the RSS subscription feed. To indicate that a feed read request was made for an RSS feed, use this parameter with `WT.rss_ev=f`.

`WT.rss_f=Sports&WT.rss_ev=f` indicates that the RSS feed name `WT.rss_f=` "sports" was the object of an RSS feed request event.

To indicate that a subscription request was made for an RSS feed, use this parameter with `WT.rss_ev=s`.

For example, `WT.rss_f=News&WT.rss_ev=s` indicates that the RSS feed name "News" was the object of an RSS subscription event.

WT.rss_a

`WT.rss_a=`*ArticleName*

This parameter identifies the RSS article. Use this parameter with `WT.rss_ev=a`.

`WT.rss_a=Global%20shortage%20of%20flu%20vaccine&WT.rss_ev=a` indicates that the RSS article name "Global shortage of flu vaccine" was the object of an article read request event.

WT.rss_ev

`WT.rss_ev=a or f or s`

This parameter identifies the RSS related event that has occurred, such as an article request `WT.rss_ev=a`, feed read request `WT.rss_ev=f`, or subscription `WT.rss_ev=s`.

WT.clip_t

`WT.clip_t=`*MediaType*

This parameter identifies the type of media that users have accessed. Use this parameter with `WT.clip_n=` and with `WT.clip_ev=`.

`WT.clip_t=Windows%20Media&WT.clip_n=Milton%20Waddams%20Presents&WT.clip_ev=v`

In this example, the clip type `WT.clip_t=` is "Windows Media," the name of the clip `WT.clip_n=` is "Milton Waddams Presents," and the event type `WT.clip_ev=v` is view.

WT.clip_n

`WT.clip_n=`*MediaClipName*

This parameter identifies the name of the clip `WT.clip_n=` that users have accessed. Typically, use this parameter with event type, `WT.clip_ev=`, and media type `WT.clip_t=`.

WT.clip_ev

`WT.clip_ev=`*EventType*

This parameter identifies the type of media-related event that has occurred, such as a view of a media event, `WT.clip_ev=v`.

WT.ria_a

`WT.ria_a=`*ApplicationName*

This parameter identifies the name of the Rich Internet Application (RIA) accessed. Typically, use this parameter with `WT.ria_c=` to identify RIA content, `WT.ria_f=` to identify the RIA feature, and `WT.ria_ev=` to identify the RIA event type.

`WT.ria_a=Homepage%20interactive%20promo%20with%20video%20and%20mp3&WT.ria_c=Vegas%20video%201&WT.ria_f=Play&WT.ria_ev=play`

In this example, the parameter identifies the RIA application as "Homepage interactive promo with video and mp3," the RIA content as "vegas," the RIA feature used as "play," and the RIA event type as "play."

WT.ria_c

`WT.ria_c=`*RIAContent*

This parameter identifies the RIA content, `WT.ria_c=`. Typically, use this parameter with `WT.ria_a=`, `WT.ria_f=`, and `WT.ria_ev=`.

WT.ria_f=

`WT.ria_f=`*RIAFeature*

This parameter identifies the RIA feature accessed, `WT.ria_f=`. Typically, use this parameter with `WT.ria_a=`, `WT.ria_c=`, and `WT.ria_ev=`.

`WT.ria_ev=`

`WT.ria_ev=`*`RIAEvent`*

This parameter identifies the RIA event that has occurred, such as the selection of a button or feature that is part of the RIA application used. Actions taken in the RIA application such as “play,” “zoom,” or “spin” are examples of an RIA event.

`WT.cgm_t=`

`WT.cgm_t=`*`CGMType`*

This parameter identifies the type of consumer generated media, `WT.cgm_t=`. Use this parameter with `WT.cgm_ev=`.

```
WT.cgm_t=Blog&WT.cgm_ev=c
```

In this example, the parameter identifies the consumer generated media type as “blog,” and the event `WT.cgm_ev=c` is a comment.

`WT.cgm_ev=`

`WT.cgm_t=`*`CGMEvent`*

This parameter identifies the consumer generated media event that has occurred, such as a post (`p`) or comment (`c`).

`WT.test_v=`

`WT.test_v=`*Variant*

This parameter identifies the test variant, `WT.test_v=` for the web site. This parameter can be populated then used to compare activity such as the scenario step conversion rate of two or more test variants.

```
WT.test_v=Product%20Layout%20B
```

In this example, the web site test variant used is “Product Layout B.”

SmartView Parameters

This section contains parameters defined specifically for use with Webtrends SmartView. For information about configuring Webtrends for SmartView, see the *SmartView User’s Guide*.

`WT.svl`

`WT.svl=`*`any string`*

For SmartView to differentiate multiple links on a web page that all lead to the same URL, use the `WT.svl` query parameter to uniquely identify the links. For example, if you have two links on your home page that both go to the store page, you should use the SmartView query parameter to identify each link.

To use the SmartView query parameter:

1. Place the `WT.svl` parameter on every page where multiple links lead to the same page. For example, `http://www.mydomain.com/?WT.svl=link1`.
2. Assign each link a unique value. SmartView uses `WT.svl` to assign the appropriate measure values to individual links.

Notes

Do not include this parameter in your URL Rebuilding definitions. Webtrends automatically recognizes this parameter and uses it only when creating SmartView custom reports. If you include `WT.svl` in a URL Rebuilding definition, non-SmartView reports are affected. If you exclude it in a URL Rebuilding definition, Webtrends won't be able to use it to differentiate links when creating SmartView reports.

`WT.tsp`

WT.tsp=1

Identifies transition source pages for SmartView. The JavaScript creates and passes this query parameter if you enabled SmartView page transition tracking in the JavaScript tag and you tagged the source page with the SmartView page transition META tag. A source page is a page that you want to be tracked for SmartView reporting. For more information, see "Configuring SmartView Using JavaScript Tags" in the *SmartView User's Guide*.

Do not use this query parameter to tag your web site. You can use this parameter to focus the analysis on only page transition pages using custom report filters.

`WT.ttp`

WT.ttp=1

Identifies transition target pages for SmartView. The JavaScript creates and passes this query parameter if you enabled SmartView page transition tracking in the JavaScript tag and you tagged the previously viewed page with the SmartView page transition META tag. For more information, see "Configuring SmartView Using JavaScript Tags" in the *SmartView User's Guide*.

Do not use this query parameter to tag your web site. You can use this parameter to focus the analysis on only page transition pages using custom report filters.

Stored Visitor Parameter

The Stored Visitor parameter identifies the unique visitor ID you assign to your visitors.

`WT.dcsvid`

`WT.dcsvid=`*`anystring`*

Webtrends Analytics Considerations

If you enable Visitor History in a profile, when Webtrends Analytics detects this parameter, it is stored in the Visitor History database. When the Visitor History database is exported, this parameter is exported along with each visitor for the purpose of identifying visitors. For more information about visitor history see "Visitor History Parameters" on page 99.

Webtrends Visitor Data Mart Considerations

Visitor Data Mart uses this parameter to populate the **`ExternalVisitorID`** field of the Visitor table, which allows you to link to external visitor data in the Extended Attributes Database. For more information about the complete set of query parameters that Visitor Data Mart uses, see "Configuring Your Web Site to Collect Visitor Data Mart Data" in the *Visitor Data Mart User's Guide*.

Visitor History Parameters

The Webtrends analysis process can generate and maintain parameters that support visitor profiling when you enable Visitor History in your profiles. These visitor-related parameters are stored in a Visitor History Table for each profile that has Visitor History enabled. Because Visitor History parameters are handled by Webtrends, you should not use these parameters in META tags on your web pages.

Parameters that describe elapsed time periods in days are calculated as a complete 24-hour period. Thus, if a visitor visits for the first time at 1:00 on Monday, then any visit before 1:00 on Tuesday is considered as zero days since the first visit (`WT.vr.fvd`), even though the actual day is different.

Most Recent Campaign Parameters

Webtrends generates and maintains these parameters when you enable Visitor History and select the Campaign History category in your profile. For more information, see "Visitor History Parameters".

`WT.vr.rac`
`WT.vr.rac=`*`MostRecentCampaign`*

Identifies the visitor's most recent campaign. Of all of the campaigns, the "most recent" campaign is the one that was most recently added to the visitor history table. This is a single value (no multiples allowed). This parameter is not set if the visitor has never had a campaign.

Unique Visitors for Campaigns

You can use the following set of parameters to track unique visitors for daily, weekly, monthly, quarterly, yearly, and lifetime campaigns. The active campaign list is used to determine the values. Therefore, you can get an additional campaign unique visitor when it is referenced again. These parameters are only available for reporting if you use the default campaign translation file. For more information, see "Lookup Tables for Drilldowns" in the *Administration User's Guide*.

`WT.vr.rac_dc`
`WT.vr.rac_dc=`*`CampaignDemandChannel`*

Identifies the campaign demand channel. It is set to the description corresponding to the ID in the translation file.

`WT.vr.rac_de`
`WT.vr.rac_de=`*`CampaignDescription`*

This parameter is set to the description corresponding to the ID in the translation file.

`WT.vr.rac_cr`
`WT.vr.rac_cr=`*`CampaignCreative`*

This parameter is set to the creative corresponding to the ID in the translation file. A campaign creative is an attribute of a specific offer, for example, a "Buy Now!" graphic. A specific offer may consist of many creatives.

WT.vr.rac_ct
`WT.vr.rac_ct=CampaignCreativeType`

This parameter is set to the creative type corresponding to the ID in the translation file.

WT.vr.rac_ma
`WT.vr.rac_ma=CampaignMarketingActivity`

This parameter is set to the marketing activity corresponding to the ID in the translation file.

WT.vr.rac_mp
`WT.vr.rac_mp=CampaignMarketingProgram`

This parameter is set to the marketing program corresponding to the ID in the translation file.

WT.vr.rac_of
`WT.vr.rac_of=CampaignOffer`

This parameter is set to the offer corresponding to the ID in the translation file.

WT.vr.rac_pa
`WT.vr.rac_pa=CampaignPartner`

This parameter is set to the partner corresponding to the ID in the translation file.

WT.vr.rac_pl
`WT.vr.rac_pl=CampaignPlacement`

This parameter is set to the placement corresponding to the ID in the translation file.

Most Recent Campaign Visitors

These parameters are generated and maintained by the Webtrends analysis process when you enable Visitor History and select the Campaign History category in your profile. For more information, see "Visitor History Parameters" on page 99.

WT.vr.rac_d
`WT.vr.rac_d=1`

Identifies a visitor's first visit for a day for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

WT.vr.rac_w
`WT.vr.rac_w=1`

Identifies a visitor's first visit for a week for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

WT.vr.rac_m
`WT.vr.rac_m=1`

Identifies a visitor's first visit for a month for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

WT.vr.rac_q
`WT.vr.rac_q=1`

Identifies a visitor's first visit for a quarter for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

WT.vr.rac_y
`WT.vr.rac_y=1`

Identifies a visitor's first visit for a year for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

`WT.vr.rac_f`
`WT.vr.rac_f=1`

Identifies a visitor's first visit for the campaign specified in `WT.mc_id` (campaign ID) on the hit.

Visitor "Initial" Parameters

The following list describes several visitor parameters that keep track of the "first" aspects of a visitor's history with the site. Note that although the parameters use the terminology "first," all preconfigured objects based on these parameters use the term "initial" (for example, Initial Referrer is the dimension based on `WT.vr.fr`). Webtrends generates these parameters when you enable Visitor History. For more information, see "Visitor History Parameters" on page 99.

`WT.vr.fr`
`WT.vr.fr=`*`FirstReferrer`*

Identifies the visitor's first recorded referrer. The format is the same as that for the Referring Domains dimension (for example, `google.com`). This is set on the first hit of the first visit and does not change afterwards.

`WT.vr.fc`
`WT.vr.fc=`*`FirstCampaign`*

Identifies the visitor's first recorded marketing campaign. The format is the same as that for the Campaign dimension. Only campaigns identified using the `WT.mc_id` parameter are counted. Campaigns defined solely through Webtrends Administration are not used.

This parameter is not provided until the visitor visits with a campaign. At that point the value and parameter are set and will never change.

`WT.vr.fe`
`WT.vr.fe=`*`FirstEntryPage`*

Identifies the visitor's first recorded page view. The format is a page URL without query parameters. This is set on the first hit of the first visit and never changes afterwards.

Elapsed Time Parameters

These parameters are generated and maintained by the Webtrends analysis process when you enable Visitor History. For more information, see "Visitor History Parameters" on page 99.

`WT.vr.fvd`
`WT.vr.fvd=`*`DaysSinceFirstVisit`*

Identifies the days since the visitor's first visit. This is an integer containing the days since the visitor's first visit. The value is truncated (for example, if 47 hours has passed since the first visit, the value is 1). This parameter is not provided on a visitor's first visit.

When using this as a measure for the Visitor dimension, use the maximum value.

When using this as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

`WT.vr.pvd`
`WT.vr.pvd=`*`DaysSincePreviousVisit`*

Days since the Visitor's Previous Visit. This is an integer containing the days since the visitor's previous visit.

This parameter makes most sense when used as a visit filter. This parameter is not provided on a visitor's first visit.

When using this parameter as a measure for the Visitor dimension, use the maximum value. When using this parameter as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

WT.vr.pvdb

`WT.vr.pvdb=`*`High/Moderate/Some/Low descriptor`*

Classifies days since the visitor's previous visit into one of four categories. This parameter is non-numeric and is used as a dimension.

You can change the number of categories by editing the `vrbucket.ini` file. By default this file is configured as follows:

```
[PVDBValues]
High = 4
Moderate = 8
Some = 12
```

The following table describes the meaning of the default values.

[PVDBValues]	Meaning
High=4	$x \leq 4$
Moderate=8	$4 < x \leq 8$
Some=12	$8 < x \leq 12$

By default, the maximum number of days for `Some` is set to 12. Any visitor whose last visit was more than 12 days ago is assigned to the Low Recency category. *Recency* is to the number of days since a visitor's most recent visit.

Example 1: Adjust all of the ranges in `vrbucket.ini` to:

```
[PVDBValues]
High = 7
Moderate = 14
Some = 21
```

Example 2: To create two buckets (for example, *`High/Low`*), change `vrbucket.ini` to:

```
[PVDBValues]
High = 4
Moderate = 4
Some = 4
```

The result of Example 2 is:

`High` is ≤ 4

`Low` is > 4

WT.vr.ppd
`WT.vr.ppd=`*`DaysSincePreviousPurchase`*

Identifies the days since the visitor's previous purchase. This is an integer value.

This parameter is typically used as a visit filter. This parameter is not provided until the visitor makes the first purchase and does change on every hit that a purchase is made after that.

When using this parameter as a measure for the Visitor dimension, use the maximum value. When using this parameter as a measure for dimensions other than Visitor, you will usually configure the average value.

The sum of this measure has no meaning.

WT.vr.lat
`WT.vr.lat=`*`VisitLatency`*

Visit Latency. The visit latency is the number of days since the visitor's first visit (`WT.vr.fvd`) divided by the number of visit intervals (`WT.vr.vc`). It gives an indication of the average elapsed time between visits. This parameter is not provided on the first visit.

When using this parameter as a measure for the Visitor dimension, use the maximum value. When using this as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

Historical Counts Parameter

Webtrends generates and maintains these parameters by the Webtrends analysis process when you enable Visitor History. For more information, see "Visitor History Parameters" on page 99.

WT.vr.vc
`WT.vr.vc=`*`VisitCount`*

Identifies the total number of visits recorded for a visitor. This is an integer representing the number of visits since the visitor's first visit. When using this as a measure for the Visitor dimension, use the maximum value. When using this as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

WT.vr.vcb
`WT.vr.vcb=`*`High/Moderate/Some/Low descriptor`*

Classifies the value of the `WT.vr.vc` parameter into one of four categories.

This parameter is non-numeric and used as a dimension. You can change the number of categories by editing the `vrbucket.ini` file. By default, the file is configured as follows:

```
[VCBValues]
High = 25
Moderate = 15
Some = 5
```

The following table shows the meaning of the default values:

[VCBValues]	Meaning
High=25	25 < *x*

[VCBValues]	Meaning
Moderate=15	$15 > x \leq 25$
Some=5	$5 > x \leq 15$

By default, the minimum value for `Some` is 5. Any visitor whose visit count value is less than 5 is assigned to the `Low` value category.

Example 1: Adjust all of the ranges in `vrbucket.ini` to:

```
[VCBValues]
High = 50
Moderate = 30
Some = 10
```

Example 2: To create two buckets (for example, *`High/Low`*), change `vrbucket.ini` to:

```
[VCBValues]
High = 25
Moderate = 25
Some = 25
```

The result of example 2 is:

`High` is `>25`

`Low` is `>0 ≤25`

Historical Transactions/Purchases Parameters

Webtrends generates and maintains these parameters by the Webtrends analysis process when you enable Visitor History. For more information, see "Visitor History Parameters" on page 99.

The following parameters are calculated using the transaction parameters. For more information, see "Transaction Parameters" on page 91.

`WT.vr.vv`

`WT.vr.vv=`*`VisitorValue`*

Webtrends generates this Visitor History parameter to track the visitor's overall value, which is the value of all purchases recorded for a visitor over time. This is a floating-point value containing the amount of money spent by this visitor back to and including the visitor's first visit.

When using this as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

`WT.vr.ltb`

`WT.vr.ltb=`*`High/Moderate/Some/Low`* `descriptor`

Webtrends generates this Visitor History parameter to classify the `WT.vr.vv` parameter value in one of four categories.

The value is non-numeric and is used as a custom report dimension.

You can change the number of categories by editing the `vrbucket.ini` file. By default, the file is configured as follows:

```
[LTBValues]
High = 750
Moderate = 500
Some = 250
```

The following table shows the meaning of the default values:

[LTBValues]	Meaning
High=750	*x* >750
Moderate=500	500 > *x* ≤750
Some=5	250> *x* ≤500

By default, the minimum value for `Some` is 250. Any visitor whose value is less than 250 is assigned to the Low category.

Example 1: Adjust all of the ranges in `vrbucket.ini` to:

```
[LTBValues]
High = 1500
Moderate = 1000
Some = 500
```

Example 2: To create two buckets (for example, *High/Low*), change `vrbucket.ini` to:

```
[LTBValues]
High = 750
Moderate = 750
Some = 750
```

The result of example 2 is:

`High` is >750
`Low` is >0 ≤750

WT.vr.ppv

`WT.vr.ppv=`*`PreviousPurchaseValue`*

Webtrends generates this Visitor History parameter to track the visitor's previous purchase amount. This is a floating-point value containing the amount spent on the most recent purchase. This parameter is not generated until a visitor makes the first purchase.

When using this parameter as a custom report measure for the Visitor dimension, use the maximum value. When using this parameter as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

WT.vr.vp

`WT.vr.vp=`*`VisitorPurchases`*

Webtrends generates this Visitor History parameter to track the total number of purchases ever made by a visitor. The value is an integer containing the number of purchase transactions (not the total number of units purchased) back to and including the visitor's first visit.

When using this parameter as a custom report measure for the Visitor dimension, use the maximum value. When using this parameter as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

WT.vr.fpd

`WT.vr.fpd=`*`DaysSinceFirstPurchase`*

Webtrends generates this Visitor History parameter to track the days since the visitor's first purchase. The value is an integer containing the days since the visitor's first purchase.

This parameter is best used as a visit filter or as a custom report measure. This parameter is not generated until a visitor makes the first purchase.

When using this parameter as a measure for the Visitor dimension, use the maximum value. When using this parameter as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

WT.vr.bpd

`WT.vr.bpd=`*`DaysBeforeFirstPurchase`*

Webtrends generates this Visitor History parameter to track the days before the visitor's first purchase. The value is an integer containing the number of days between the visitor's first visit and the first purchase.

This parameter is best used as a visit filter or as a measure. This parameter is not provided until a visitor makes the first purchase.

When using this as a measure for dimensions other than Visitor, you usually configure the average value.

The sum of this measure has no meaning.

Webtrends generates the following set of parameters to track unique buyers and buyer's status for daily, weekly, monthly, quarterly, yearly, and lifetime periods.

WT.vr_brws

`WT.vr_brws=`*`Buyer/Non-Buyer`*

Webtrends generates this query parameter on a visitor's first visit of the day. Its value indicates whether a visitor has purchased in the past. `Buyer` indicates that the visitor has purchased before. `Non-buyer` indicates that the visitor has not purchased.

A visitor is considered a buyer if the `WT.tx_s` parameter is passed in the query string for the visit. A visitor is considered a non-buyer if `WT.tx_s` is not passed in the query string. In addition, Webtrends uses the following invoice parameters to evaluate whether `WT.tx_s` should be used for buyer determination:

- `WT.tx_id` and `WT.tx_it`
- `WT.tx_i`

If both `WT.tx_id` and `WT.tx_it` parameters are passed during the visit and are properly formatted, Webtrends uses them to evaluate whether `WT.tx_s` should be used for buyer determination. If the date and time specified by these parameters is older than the time of the hit by more than the configured invoice age limit, `WT.tx_s` is not used and the visitor is considered a non-buyer. The invoice age limit is set at two days, meaning that any invoices three days or older than the first hit associated with the invoice are not used for buyer determination.

If the `WT.tx_i` parameter is passed during the visit, Webtrends uses it to evaluate whether `WT.tx_s` should be used for buyer determination. Webtrends looks for the invoice number passed in for this query parameter in the visitor's invoice history. If it has seen this invoice number before, `WT.tx_s` is not used and the visitor is considered a non-buyer. Three invoices are kept per visitor. Invoices more than two days old are purged from the visitor's invoice history.

WT.vr_by
`WT.vr_by=*New Buyer/Repeat Buyer*`

Webtrends generates this Visitor History parameter when a visitor makes a purchase. It indicates whether the visitor is purchasing for the first time or has purchased before.

WT.vr.bt_d
`WT.vr.bt_d=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first daily purchase. This parameter is generated and set to 1 for the first purchase from a visitor for a day.

WT.vr.bt_w
`WT.vr.bt_w=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first weekly purchase. This parameter is generated and set to 1 for the first purchase from a visitor during a week.

WT.vr.bt_m
`WT.vr.bt_m=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first monthly purchase. This parameter is generated and set to 1 for the first purchase from a visitor during a month.

WT.vr.bt_q
`WT.vr.bt_q=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first quarterly purchase. This parameter is generated and set to 1 for the first purchase from a visitor during a quarter.

WT.vr.bt_y
`WT.vr.bt_y=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first yearly purchase. This parameter is generated and set to 1 for the first purchase from a visitor during a year.

WT.vr.bt_f
`WT.vr.bt_f=*1*`

Webtrends generates this Visitor History parameter to identify the visitor's first purchase. This parameter is generated and set to 1 for the first purchase from a visitor.

Search Engine Parameters

The Search Engine parameters keep track of initial and most recent search engines and search engine phrases for a visitor. These parameters work in conjunction with the `WT.srch` parameter to determine whether the referring search engine was from a paid search phrase. Webtrends generates and maintains these parameters when you enable Visitor History and select the Search Engine History category in your profile. For more information, see "Visitor History Parameters" on page 99.

WT.vr.ise
`WT.vr.ise=*InitialSearchEngine*`

Webtrends generates this Visitor History parameter to track the visitor's initial search engine. This parameter contains the string identifying the initial search engine for a visitor. The parameter is generated with the hit where it is recognized as the referring site. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.isep

`WT.vr.isep=`*`InitialSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's initial search engine phrase. This parameter contains the string identifying the initial search engine phrase for a visitor. The parameter is generated with the hit where the search engine is recognized as the referring site. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.ipd_se

`WT.vr.ipd_se=`*`InitialPaidSearchEngine`*

Webtrends generates this Visitor History parameter to track the visitor's initial paid search engine. This parameter contains the string identifying the initial paid search engine for a visitor. A paid search engine referrer is identified by a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.ipd_se` parameter is provided with the hit where it is recognized as the referring site. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.ipd_sep

`WT.vr.ipd_sep=`*`InitialPaidSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's initial paid search engine phrase. This parameter contains the string identifying the initial paid search engine phrase for a visitor. A paid search engine referrer/phrase is identified by a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.ipd_sep` parameter is provided with the hit where it `WT.vr.ipd_se` is recognized. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is first recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.iog_se

`WT.vr.iog_se=`*`InitialOrganicSearchEngine`*

Webtrends generates this Visitor History parameter to track the visitor's initial organic search engine. This parameter contains the string identifying the initial organic search engine for a visitor. An organic search engine referrer is identified by the lack of a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.iog_se` parameter is provided with the hit where it is recognized as the referring site. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.iog_sep

`WT.vr.iog_sep=`*`InitialOrganicSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's initial organic search engine phrase. This parameter contains the string identifying the initial paid search engine phrase for a visitor. An organic search engine referrer/phrase is identified by the lack of a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.iog_sep` parameter is provided with the hit where it `WT.vr.iog_se` is recognized. The value of the parameter never changes and is provided with the first hit of every visit after the visit in which it is first recognized. The parameter is not provided until it has been recognized and set to the initial value.

WT.vr.r_se

`WT.vr.r_se=`*`MostRecentSearchEngine`*

Webtrends generates this Visitor History parameter to track the visitor's most recent search engine. This parameter contains the string identifying the most recent search engine for a visitor. The `WT.vr.r_se` parameter is provided with the hit where it is recognized as the referring site. The value of the parameter changes whenever a new search engine is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

WT.vr.r_sep

`WT.vr.r_sep=`*`MostRecentSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's most recent search engine phrase. This parameter contains the string identifying the most recent search engine phrase for a visitor. The `WT.vr.r_sep` parameter is provided with the hit where `WT.vr.r_se` is recognized as the referring site. The value of the parameter changes whenever a new search engine/search engine phrase is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

WT.vr.rpd_se

`WT.vr.rpd_se=`*`MostRecentPaidSearchEngine`*

Webtrends generates this Visitor History parameter to track the visitor's most recent paid search engine. This parameter contains the string identifying the most recent paid search engine for a visitor. A paid search engine referrer/phrase is identified by a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.rpd_se` parameter is provided with the hit where it is recognized as the referring site. The value of the parameter changes whenever a new search engine is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

WT.vr.rpd_sep

`WT.vr.rpd_sep=`*`MostRecentPaidSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's most recent paid search engine phrase. This parameter contains the string identifying the most recent paid search engine phrase for a visitor. A paid search engine referrer/phrase is identified by a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.rpd_sep` parameter is provided with the hit where `WT.vr.rpd_se` is recognized as the referring site. The value of the parameter changes whenever a new search engine/search engine phrase is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

WT.vr.rog_se

`WT.vr.rog_se=`*`MostRecentOrganicSearchEngine`*

Webtrends generates this Visitor History parameter to track the visitor's most recent organic search engine. This parameter contains the string identifying the most recent organic search engine for a visitor. An organic search engine referrer/phrase is identified by the lack of a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.rog_se` parameter is provided with the hit where it is recognized as the referring site. The value of the parameter changes whenever a new search engine is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

WT.vr.rog_sep

`WT.vr.rog_sep=`*`MostRecentOrganicSearchEnginePhrase`*

Webtrends generates this Visitor History parameter to track the visitor's most recent organic search engine phrase. This parameter contains the string identifying the most recent organic search engine phrase for a visitor. An organic search engine referrer/phrase is identified by the lack of a `WT.srch=1` parameter in the query field of hit query string. The `WT.vr.rog_sep` parameter is provided with the hit where `WT.vr.rog_se` is recognized as the referring site. The value of the parameter changes whenever a new search engine/search engine phrase is recognized. It is provided with the first hit of every visit after the visit in which it is recognized. It changes whenever a new search engine is recognized. The parameter is not provided until it has been recognized and set to a first value.

Visitor Tracking Parameters

Visitor Tracking parameters let you track daily, weekly, monthly, quarterly, yearly, and all-time unique visitors. Webtrends generates and maintains these parameters when you enable Visitor History and select the Visit History category in your profile. For more information, see "Visitor History Parameters" on page 99.

WT.vr.vt_d

`WT.vr.vt_d=`*`1`*

Webtrends generates this Visitor History parameter to track daily visitor daily activity. This parameter is present and set to 1 on a new visitor's first hit for the day.

WT.vr.vt_w

`WT.vr.vt_w=`*`1`*

Webtrends generates this Visitor History parameter to track weekly visitor activity. This parameter is present and set to 1 on a new visitor's first hit for the week.

WT.vr.vt_m

`WT.vr.vt_m=`*`1`*

Webtrends generates this Visitor History parameter to track monthly visitor activity. This parameter is present and set to 1 on a new visitor's first hit for the month.

WT.vr.vt_q

`WT.vr.vt_q=`*`1`*

Webtrends generates this Visitor History parameter to track quarterly visitor activity. This parameter is present and set to 1 on the first hit from a new visitor during a quarter.

WT.vr.vt_y

`WT.vr.vt_y=`*`1`*

Webtrends generates this Visitor History parameter to track yearly visitor activity. This parameter is present and set to 1 on the first hit from a new visitor during a year.

WT.vr.vt_f
`WT.vr.vt_f=1`

Webtrends generates this Visitor History parameter to track a visitor's first hit. This parameter is present and set to 1 on the first hit from a new visitor.

WT.vr.piv_d
`WT.vr.piv_d=1`

Webtrends generates this Visitor History parameter to track daily page of interest activity. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest during a day.

WT.vr.piv_w
`WT.vr.piv_w=1`

Webtrends generates this Visitor History parameter to track weekly page of interest activity. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest during a week.

WT.vr.piv_m
`WT.vr.piv_m=1`

Webtrends generates this Visitor History parameter to track monthly page of interest activity. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest during a month.

WT.vr.piv_q
`WT.vr.piv_q=1`

Webtrends generates this Visitor History parameter to track quarterly page of interest activity. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest during a quarter.

WT.vr.piv_y
`WT.vr.piv_y=1`

Webtrends generates this Visitor History parameter to track yearly page of interest activity. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest during a year.

WT.vr.piv_f
`WT.vr.piv_f=1`

Webtrends generates this Visitor History parameter to track a visitor's first page of interest. This parameter is present and set to 1 on the first hit from a new visitor to a page of interest.

WT.vr.cgv_d
`WT.vr.cgv_d=1;...`

Webtrends generates this Visitor History parameter to track daily content group activity. This parameter is present and set to 1 for the first hit from a new visitor to a content group during a day. If multiple content groups are specified on the hit, this parameter contains as many values as there are content groups.

WT.vr.cgv_w
`WT.vr.cgv_w=1;...`

Webtrends generates this Visitor History parameter to track weekly content group activity. This parameter is present and set to 1 for the first hit from a new visitor to a content group during a week. If multiple content groups are specified on the hit, this parameter contains as many values as there are content groups.

WT.vr.cgv_m
`WT.vr.cgv_m=1;...`

Webtrends generates this Visitor History parameter to track weekly content group activity. This parameter is present and set to 1 for the first hit from a new visitor to a content group during a month. If multiple content groups are specified on the hit, this parameter contains as many values as there are content groups.

WT.vr.cgv_q
WT.vr.cgv_q=*1;...*

Webtrends generates this Visitor History parameter to track quarterly content group activity. This parameter is present and set to 1 for the first hit from a new visitor to a content group during a quarter.

WT.vr.cgv_y
WT.vr.cgv_y=*1;...*

Webtrends generates this Visitor History parameter to track yearly content group activity. This parameter is present and set to 1 for the first hit from a new visitor to a content group during a year. If multiple content groups are specified on the hit, this parameter contains as many values as there are content groups.

WT.vr.cgv_f
WT.vr.cgv_f=*1;...*

Webtrends generates this Visitor History parameter to track a visitor's first hit to a content group. This parameter is present and set to 1 for the first hit from a new visitor to a content group. If multiple content groups are specified on the hit, this parameter contains as many values as there are content groups.

Visitor Segmentation Parameters

Visitor Segmentation parameters allow you to store the most recent value of a segmentation query parameter for inclusion in your reports. For example, your travel web site tracks visitors using a segmentation parameter, such as `WT.seg_1`, to identify the visitor's "traveler type." The result of the visitor's most recent traveler type value is stored in the `WT.vhseg_1` parameter in the Visitor History database. Custom reports you create that use the `WT.vhseg_1` parameter as a dimension show statistics for the most recent value of the key parameter.

The most recent value of the key parameter, `WT.seg_`*x* is stored in the corresponding result parameter. To report on visitor segmentation data, create a custom report that uses your result parameter as a dimension.

Webtrends generates and maintains these parameters when you enable Visitor History and select the Custom Visitor Segmentation category in your profile. You must also implement the `WT.seg` parameter on your web pages. For more information about `WT.seg`, see "Segment Parameter" on page 94.

WT.vhseg_1
WT.vhseg_1=*VisitorSegment1Result*

WT.vhseg_2
WT.vhseg_2=*VisitorSegment2Result*

WT.vhseg_3
WT.vhseg_3=*VisitorSegment3Result*

WT.vhseg_4
WT.vhseg_4=*VisitorSegment4Result*

SDC-Generated Visitor Parameters

The following subsections discuss visitor-related parameters that are generated and maintained by SmartSource Data Collector (SDC).

Visitor Tracking Parameters

Visitor Tracking parameters allow you to track daily, weekly, monthly, quarterly, and yearly unique visitors. SDC inserts these parameters into the cs-uri-query strings.

WT.vt_tlv

`WT.vt_tlv=*UNIX Time*`

SDC generates this parameter to identify the time of the visitor's last visit. The value is expressed as the number of seconds since 1970 (standard UNIX time), which is calculated using information stored in the third-party cookie value. SDC only sets this parameter at the start of a new visit. On a visitor's first visit, the value is set to zero. If you disable cookie tracking, this parameter is not generated or passed in the query string.

Visitor Data Mart uses this query parameter to determine whether a new visit should also be counted as a new daily, weekly, monthly, quarterly, or yearly visitor. For example, if the day of the new visit is different than the day of the previous visit, relative to the GMT offset of the JavaScript tag, the visit is counted as a new daily visit.

WT.vt_f_tlv

`WT.vt_f_tlv=*UNIX Time*`

SDC generates this parameter to identify the time of the visitor's last visit. The value is expressed as the number of seconds since 1970 (standard UNIX time), which is calculated using information stored in the first-party cookie value. SDC only sets this parameter at the start of a new visit. On a visitor's first visit, the value is set to zero. If you disable first-party cookie tracking in the JavaScript tag, this parameter is not generated or passed in the query string.

Visitor Data Mart uses this query parameter to determine whether a new visit should also be counted as a new daily, weekly, monthly, quarterly, or yearly visitor. For example, if the day of the new visit is different than the day of the previous visit, relative to the GMT offset of the JavaScript tag, the visit is counted as a new daily visit.

WT.vt_d

`WT.vt_d=1`

SDC generates this parameter to track daily visitors for Express Analysis. This parameter is generated and set to 1 for the first hit from a new visitor for a day.

WT.vt_a_d

`WT.vt_d=1`

SDC generates this parameter to track daily visitors for Account Rollup Data Sources. This parameter is generated and set to 1 for the first hit from a new visitor for a day for a given account. This parameter is used in conjunction with Account Rollup Profiles.

WT.vt_f_d

`WT.vt_f_d=1`

SDC generates this parameter to track daily visitors for Express Analysis only. This parameter is generated and set to 1 for the first hit from a new visitor when you configure your SmartSource Data Source to use the First-Party Cookie JavaScript. The First-Party Cookie JavaScript generates this parameter and its value.

WT.vt_s

`WT.vt_s=1`

SDC generates this parameter to track visitor sessions for Express Analysis and Visitor Data Mart.

This parameter is generated and set to 1 for the first hit for a new session. Cookie tracking must be enabled to set this query parameter.

WT.vt_a_s
`WT.vt_a_s=1`

SDC generates this parameter to track visitor sessions for Account Rollup Data Sources. Applies to Real Time analysis only. This parameter is generated and set to 1 for the first hit for a new session for a given account. This parameter is used in conjunction with Account Rollup Profiles.

WT.vt_f_s
`WT.vt_f_s=`*`1`*

SDC generates this parameter to track new visitor sessions for Express Analysis and Webtrends Visitor Data Mart.

This parameter is generated and set to 1 for the first hit for a new session. The First-Party Cookie JavaScript generates this parameter.

WT.vt_f
`WT.vt_f=`*`1`*

SDC generates this parameter to track new and returning visitors. This parameter is generated and set to 1 for the first hit from a new visitor. This parameter is set to 2 if the visitor's browser does not accept cookies. The First-Party Cookie JavaScript generates this parameter.

WT.vt_f_a
`WT.vt_f_a=`*`1`*

SDC generates this parameter to track new visitors for an account. This parameter is generated and set to 1 for the first hit from a new visitor for a given account. This parameter is set to 2 if the visitor's browser does not accept cookies. The First-Party Cookie JavaScript generates this parameter. This parameter is used in conjunction with Account Rollup Profiles.

WT.vt_sid
`WT.vt_sid=`*`identifier`*

SDC generates this parameter to identify visitor sessions for Visitor Data Mart. This parameter is generated on every hit to identify the visitor session.

This identifier is formed by concatenating two pieces of data:

- *Identifier*: The value of `WT.co_f`. If `WT.co_f` is not present on the incoming hit, SDC generates this value. This is the unique identifier that is generated at the time of a new visit. It is 32-character hexadecimal number (0-9, a-f)
- *Timestamp*: The time when the session began. This is the number of milliseconds since January 1, 1970.

Note

The First Party Cookie Javascript no longer emits `WT.vt_sid`. Rather, it now emits the two components (*Identifier* and *timestamp*) as separate parameters. See `WT.vtid` and `WT.vtvs` for more information.

Example:

```
WT.vt_sid=10.61.19.29-3899933120.29768655.1141069
```

WT.vtid
`WT.vtid=`*`identifier`*

This parameter is used to identify visitors for Visitor Data Mart. By default, this identifier is provided by Webtrends Data Collection Servers using an in-line JavaScript request for `wtid.js` at the time of a new visit. Alternatively, this parameter value can come from a custom query parameter or customer cookie.

Identifier: Any alphanumeric string that uniquely identifies a visitor. By default this value is the same as the `WT.co_f` parameter.

WT.vtvs

`WT.vtvs=`*`timestamp`*

The JavaScript tag generates this parameter to identify visitor sessions for Webtrends Visitor Data Mart. This parameter is generated on every hit to identify the visitor session.

Timestamp: The time when the session began. This is the number of milliseconds since January 1, 1970.

Cookie Detection Parameters

This set of parameters allows Webtrends Analytics to tie a visitor's first hit with the rest of the visitor session. These parameters are generated and maintained by SmartSource Data Collector (SDC).

WT.co

`WT.co=`*`Yes/No`*

SDC generates this parameter to determine whether the visitor's browser supports and is configured to accept cookies. Valid values are `Yes` and `No`.

Example:

```
var coQueryParam = "&WT.co=" + navigator.cookieEnabled() ? "Yes" : "No";
```

WT.co_d

`WT.co_d=`*`Cookie_Data`*

SDC generates this parameter the first time it attempts to set the cookie. The value is set to the value of the cookie. This parameter is generated only for "first visit" hits.

Webtrends Analytics and Visitor Data Mart use this parameter for "session stitching." The first hit of the first visitor session (which may not have a cookie) gets the cookie value in the `WT.co_d` parameter. Subsequent hits that have the same value for the Webtrends cookie can be tied together with the `WT.co_d` hit to form a complete picture of the session. Also, SDC passes the `WT.co_d` value; it is not passed from the visitor's web browser.

The Webtrends cookie format contains the IP address of the cookie's connection address and the creation time. The creation time is represented as the number of seconds and nanoseconds since 1970 (standard UNIX time). A checksum is appended to the cookie value. The following example shows the format of the cookie value:

`WEBTRENDS_ID=`*`IP Address-SSSSSSSSS.NNNNNN`*`::`*`checksum`*

The following example shows the `WT.co_d` parameter with a Webtrends cookie value:

```
WT.co_d=192.168.100.40-1045156016.29542554::A2D3FC34517CE562A9D4E33EF85D7B7F
```

WT.co_a

`WT.co_a=`*`Cookie_Data`*

SDC generates this parameter the first time it attempts to set the account cookie. Note that this parameter is generated only for hits on the first visit. This parameter is used to track visitor sessions across multiple accounts in Webtrends Analytics On Demand or multiple SmartSource data sources in Webtrends Analytics software.

This parameter is generated and the value is set to the account rollup cookie's value if SDC attempted to set a first-time cookie for a given account. The global rollup cookie is named `ACOOKIE`. The global rollup cookie contains an encoded account rollup cookie named `WT_ACCT`. It contains the IP address of the cookie's connection address and the creation time. The creation time is represented as the number of seconds and nanoseconds since 1970 (standard UNIX time). A checksum is appended to the cookie value. The following example shows the format of the cookie value:

`WT_ACCT=`*`IP Address-SSSSSSSSS.NNNNNN::checksum`*

The following example show the `WT.co_a` parameter with an account rollup cookie:

```
WT.co_a=192.168.100.40-1045156016.29542554
```

`WT.co_f`

`WT.co_f=`*`uniqueIdentifier`*

SDC generates this parameter when you enable first-party cookie tracking in the Webtrends JavaScript tag. This parameter is passed on every hit so that Webtrends Analytics and Visitor Data Mart can use it for visitor session tracking.

A unique identifier is passed as the value of the `WT.co_f` query parameter. The format of the unique identifier depends on the first-party cookie tracking method you specified in the SmartSource Data Source settings.

You can configure the method that you want to use for your first party cookies in the SmartSource Data Source settings. For more information about first-party cookie tracking methods, see .

URL Truncation Parameter

SDC uses this parameter to overcome a maximum URL length limitation imposed by Internet Explorer (Microsoft Knowledge Base Article – 208427). The URL length must be 2048 or less. If the Webtrends JavaScript tag generates a URL in excess of 2048 characters and the client browser is Microsoft Internet Explorer, the hit is truncated and is passed to SDC.

`WT.tu`

```
WT.tu=1
```

SDC generates this parameter and sets it to 1 if the URL was deemed too long and truncated by the JavaScript tag. If present, SDC writes an error and discards the hit.

You can configure the `logtruncatedhits` setting to log the truncated hit rather than discard it.

HTTP Headers

You may want to access custom HTTP request headers. These headers can be inserted by third-party products such as load balancers, application servers, or web server plug-ins. This parameter is assigned the value of the specified HTTP header, which can then be referenced in a Webtrends custom report.

`WT.hdr.`*`HTTP Header`*

`WT.hdr.`*`HTTP Header=Value`*

If the header is present in the incoming request, the header name is appended to `WT.hdr`. and the header value is assigned to the value. For example, suppose that a customer wants to log the `Accept:` header, and it comes in as `Accept: */*`. The resultant parameter would be `WT.hdr.Accept=*/*`. Note that values are URL encoded.

JavaScript Tag Version

WT.tv

WT.tv=*major.minor.revision*

The JavaScript tag contains this parameter, which specifies the version of the Webtrends JavaScript tag that is currently deployed. The value is passed as *major.minor.revision* where *major.minor* specifies the Webtrends Analytics version and *revision* specifies the version of the JavaScript tag.

Although this parameter is not used in reports, it can be useful for Support when troubleshooting a tagging problem.

Site ID

WT.site

WT.site=siteId

This Visitor Data Mart parameter enables the web site owner to supply a site ID for one or more events when multiple events occur on the same hit. The JavaScript tag generates and sets this parameter to the domain name/Site Id for cross-domain Visitor Data Mart use, enabling users to select only events for a certain ID when querying the database.

Currently, this parameter is not supported in Tag Builder and must be configured manually.

Traffic Source

WT.tsrc

WT.tsrc=*traffic source value* (overrides default value)

By default, Webtrends reports on the source of traffic based on its referrer using the following values: "Paid Search," "Non-Search Campaigns," "Organic Search," "Other Referrers," or "Direct". This information is tracked in the custom Traffic Source dimension. You can use the WT.tsrc parameter to override any of the default Traffic Source values with a custom value.

Event Tracking

WT.dl

WT.dl=*event id*

Specifies the kind of event tracked.The WT.dl parameter passes a set of identifiers. Each identifier is associated with a given event which is typically mouse related. Identifiers are numeric, and are assigned inside an event handler. The WT.dl parameter can be used to filter event related traffic.

The JavaScript tag generates this parameter with the appropriate event id as shown in the following table:

Event Id	Event	Description
0	Page View	Generated when page is loaded.

Event Id	Event	Description
20	Download	Generated when a download link is clicked. Download links are onsite links whose file type matches a configurable list of download file types.
21	Anchor	Generated when an anchor link is clicked. Anchor links are on-site links that contain anchor text.
22	Dynamic "javascript"	Generated when a link containing a JavaScript URL is clicked. Example: `<a href="javascript:alert('Good Morning')">Good Morning</a>`
23	Dynamic "mailto:”	Generated when a link containing a mailto URL is clicked. Example: `<a href="mailto:firstname.lastname@company.com?subject=Good%20Morning">Send Email</a>`
24	Off-site	Generated when an off-site link is clicked. Off-site links are page elements that lead to web sites that are not instrumented with the customers SmartSource tags. The list of on-site domains is user-configurable in Webtrends Administration.
25	Right-click	Generated when a download link is right-clicked. Download links are on-site links whose file type matches a configurable list of download file types. The list of download file types is user-configurable in Webtrends Administration.
26	Form Button - Get method	Generated when a form button is clicked. Button is enclosed inside a form, and the method is GET.
27	Form Button - Post method	Generated when a form button is clicked. Button is enclosed inside a form, and the method is POST.
28	Form Button - Input tag	Generated when a form button is clicked. Button is not enclosed inside a form, but is enclosed in an <input> tag.
29	Form Button - Button tag	Generated when a form button is clicked. Button is not enclosed inside a form, but is enclosed in a `<button>` tag

Event Id	Event	Description
30	Image Map	Generated when an image map is clicked.

Parent DIV/Table ID

WT.nv

`WT.nv=<id||class>`

This parameter contains the id or class of the parent DIV or TABLE of the element that was clicked. The `WT.nv` value allows you to see the areas on a page to which an element was clicked belongs.

Click-Based Tracking

WT.es

`WT.es=hostname/page`

This parameter permits tracking click-based events and the source page from which they originated. The JavaScript tag forms this parameter by concatenating the `dcssip` and `dcsuri` parameters. to determine what page a user was on when an "event" occurred. For example, when a user clicks on a link, the event source shows what page the user was viewing when the click occurred. Although in the case of a page view, these URLs are identical, in the case of a click the dcsuri is actually the "destination URL" of the click, while the event source is the "source URL" of the click.

DCSID

WT.dcs_id

`WT.dcs_id=DCSID`

This parameter contains the value of the DCSID that generated the hit. The Standard Analysis Engine, Express Analysis Engine, and Event Database Loader pass the DCSID as value of for this parameter. This parameter becomes most useful when tracking multiple DCSIDs in SmartSource files from multiple sites. In this case, you can use this parameter to segment your report data by site. For example, you can use this parameter as a dimension in a custom report to report on activity for each site.

Note

If you look at your SmartSource files, you will not find this parameter in the query string. This is because Webtrends Analytics adds it to the query parameter set during analysis.

SDC-Parameter Override Parameters

You can use the parameters in this section to override SDC parameters on the client side. Consider the following example:

If you want a specific page, `/xyz.html`, logged to the `cs-uri-stem` field, you can assign the page name to the `dcsuri` parameter in the JavaScript tag as shown in the Modified Tag.

Default JavaScript Tag:

```
DCS.dcsuri="window.location.pathname;
```

Modified JavaScript Tag:

```
DCS.dcsuri="/xyz.html";
```

However, because modifying JavaScript is error-prone, you could instead use the `DCS.dcsuri` parameter in a META tag to override the `dcsuri` assignment in the JavaScript tag. Your META tag would look like this:

```
<META NAME="DCS.dcsuri" CONTENT="/xyz.html">
```

Keep in mind that because these parameters simply override assignments in the JavaScript tag, the parameters themselves are not actually sent to SDC. The JavaScript tag contains a custom object named `DCS`. This object contains property name/value pairs that are used to form query parameters sent to SDC. To continue our example, the JavaScript tag first extracts the META tag information and performs the following assignment:

```
DCS.dcsuri=/xyz.html
```

Next, the JavaScript tag iterates through all name/value pairs in the DCS object and forms query parameters.

In our example, the following query parameter is formed:

```
&dcsuri=/xyz.html
```

Note that the custom object name itself (DCS) is not sent to SDC.

`DCS.dcsref`
`DCS.dcsref=`*`Referrer`*

This parameter is assigned to the `dcsref` parameter before the hit is sent to SDC. The value is included in the `cs(Referer)` field of the log file.

`DCS.dcssip`
`DCS.dcssip=`*`Domain`*

This parameter is assigned to the `dcssip` parameter before the hit is sent to SDC. The value is included in the `cs-host` field of the log file.

`DCS.dcsua`
`DCS.dcsua=`*`user agent`*

This parameter is assigned to the `dcsua` parameter before the hit is sent to SDC. Value is included in the `cs(user agent)` field. Use a plug sign to encode spaces rather than %20.

`DCS.dcsuri`
`DCS.dcsuri=`*`uri-ste`*`m`

This parameter is assigned to the `dcsuri` parameter before the hit is sent to SDC. The value is included in the `cs-uri-stem` field of the log file.

`DCS.dcspro`
`DCS.dcspro=`*`Protocol`*

This parameter is assigned to the `dcspro` parameter before the hit is sent. The value is included in the `cs(Version)` field of the log file.

`DCS.dcsqry`
`DCS.dcsqry=`*`uri-query`*

This parameter is assigned to the dcsqry parameter before the hit is sent. The value is included in the `cs-uri-query` field of the log file.

DCS.dcsaut
DCS.dcsaut=*authenticated username*

This parameter is assigned to the dcsaut parameter before the hit is sent to SDC. The value is included in the cs-username field.

DCS.dcsmet
DCS.dcsmet=*method*

This parameter is assigned to the dcsmet parameter before the hit is sent to SDC. The value is included in the cs-method field.

DCS.dcssta
DCS.dcssta=*status*

This parameter is assigned to the dcssta parameter before the hit is sent to SDC. The value is included in the sc-status field.

DCS.dcsbyt
DCS.dcsbyt=*bytes*

This parameter is assigned to the dcsqry parameter before the hit is sent to SDC. The value finally is included in the sc-bytes field.

DCS.dcscip
DCS.dcscip=*ip address*

This parameter is assigned to the dcscip parameter before the hit is sent to SDC. Value is included in the c-ip field.

DCS.dcsua
DCS.dcsua=*user agent*

This parameter is assigned to the dcsua parameter before the hit is sent to SDC. Value is included in the cs(user agent) field. Use a plug sign to encode spaces rather than %20.

Conversion Plug-In Parameters

The Webtrends encoding conversion plug-in uses the parameters in this section during the character encoding conversion process. The Webtrends JavaScript tag generates these parameters when the gi18n global variable is set to true and when a web page contains DCSext query parameters.

For more information about DCSext query parameters, see "Customizing the Webtrends JavaScript Tag" in *Webtrends Analytics On Demand Implementation Guide*. For more information about the encoding conversion plug-in see "Internationalization and Webtrends" in the *Administration User's Guide*.

WT.dep
WT.dep=*DCSext parameter1[;DCSext parameter2...]*

Contains a semicolon-delimited list of the custom DCSext query parameters on a web page. The encoding conversion plug-in uses this information to identify the parameters that are known to be encoded in UTF-8. For example, a web page that contains DCSext.abc=655 and DCSext.xyz=889 would be captured by the JavaScript tag as WT.dep=abc;xyz.

Note
Do not use this parameter for collecting data.

Mobile Web and Mobile Application Parameters

These parameters are used with the Webtrends Client Libraries for Mobile Applications to track both web and application activity associated with mobile devices. These values have a maximum length of 512 characters. For more information about using these parameters with the Mobile Libraries, visit developer.webtrends.com.

Application Ad Click

WT.a_ac

WT.a_ac=*1*

Indicates that an ad was clicked within a mobile application.

Application Ad Impression

WT.a_ai

WT.a_ai=*1*

Indicates that one or more ads was viewed in a mobile application. Typically, this parameter is used with the WT.a_an parameter to specify one or more ads viewed in a single application screen.

Application Ad Name

WT.a_an

WT.a_an=*Name[;…]*

Specifies one more names of ads presented in a mobile application. You can pass multiple ad names using semicolons to delimit the list. The maximum length for each *Name* is 64 bytes.

Application Name

WT.a_nm

WT.a_nm=*application name*

Specifies the name of a mobile application. This parameter can be used to report on the usage of different applications.

Application Category

WT.a_cat

WT.a_cat=*application category*

Specifies a category of mobile application such as Games, Health & Wellness, or Productivity. This parameter can be used to report on the usage of different types of apps.

Application Publisher

WT.a_pub

WT.a_pub=*application publisher*

Specifies the developer, publisher, or vendor for an application. This parameter is required.

Connection Type

WT.ct

`WT.ct=connectionType`

Specifies the connection type used to send the data. Many mobile devices go in and out of various connection states such as wifi, 3G, Edge, or 7.1G. To avoid sending a flood of this data as the connection fluctuates, the best practice is to send this parameter only once per session. The Webtrends Mobile Application Libraries use this best practice.

Country

WT.g_co

`WT.g_co=`*`Country Code`*

Specifies the country of origin for mobile traffic using a country code.

Device Model

WT.dm

`WT.dm=`*`device model`*

Specifies the model of the mobile device that an application is running on, for example the model of a mobile phone. For example, `WT.dm=Motorola Droid` or `WT.dm=iPhone 3GS`.

Event Time Stamp

WT.ets

`WT.ets=`*`epoch time stamp`*

Specifies an epoch (unix) time stamp for an event. This time stamp is recorded so events that occur when a device is offline can be sent as an ordered batch when the device comes back online. For example, For example, `WT.ets=1006725234`.

Event Type

WT.ev

`WT.ev=`*`eventType`*

Specifies the type of event activity on a mobile device. For example, this parameter can be used to differentiate activities such as clicks, swipes, and content views.

Geolocation Coordinates

WT.gc

`WT.gc=`*`latitude`*`,`*`longitude`*

Specifies the latitude and longitude of a mobile device.

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2010 Release	November 2010	Added more parameters to support new Mobile Report Pack.
Spring 2010 Release	February 2010	Added “Mobile and Application Parameters.”
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	• Added Document Revision History section. • Corrected event ID typo’s (for Dynamic “Mail To” and “Off-site” events) for “WT.dl” on page 117

Chapter 10

Securing Your Implementation

This chapter discusses methods for securing your Webtrends implementation.

Configuring Security for Webtrends On Demand

You can configure security for Webtrends On Demand accounts by managing the security options for your account and by emphasizing the importance of secure passwords.

In Webtrends Accounts, you can perform several security enhancing tasks:

- Reduce the vulnerability of passwords by enforcing rules for users.
- Increase account security by locking out inactive users or those who have made too many login attempts.
- Increase the security of your web analytics data by using SSL to secure the entire user session.
- Decrease the risk of exposing user login credentials by preventing users from saving login information to their local computer.

The Importance of Password Security

Secure passwords are important because they ensure that only authorized users access your accounts. When you or a user creates or changes a password, it must meet the following requirements:

- Must contain at least 6 characters. As a best practice, specify a password that is at least 8 characters.
- Must contain both upper and lower case letters (a-z, A-Z).
- Must contain at least one number (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
- Cannot be part of your login user name.
- Cannot contain your first or last name.
- Cannot contain spaces.
- Must be no more than 20 characters in length.

Best Practices for Webtrends On Demand Security

As a Webtrends administrator, you should routinely perform several security-related maintenance tasks:

- Create a user account for each Webtrends user. Webtrends provides auditing tools that enable administrators to track logins and configuration changes. These tools provide the most insight when Webtrends can identify users individually. To prevent a user account from being misused, do not create a general user account that a group of users can share.
- Review user accounts periodically. Investigate inactive user accounts and consider deletion.
- Delete accounts of users when they leave the organization.

Configuring Webtrends On Demand Security

You can harden Webtrends On Demand accounts to reduce vulnerabilities by taking advantage of the security options available in Webtrends On Demand.

To access security options:

1. In the left pane of the Account Console, click **Accounts** > **Security Options**.
2. Configure the security options to your preferences.
3. Use the Help for details about this dialog.

Document Revision History

Table 1: Document Revision History contains a summary of changes made to this document beginning with the release of Webtrends Analytics, version 8.7.

Table 1: Document Revision History

Software Version	Date of Last Update	Summary of Changes
Fall 2009 Release	November, 2009	Corporate Branding Changes
v8.7d	July, 2009	Added footer link to Documentation Center.
v8.7	March, 2009	Added Document Revision History section.

Index