



THE  
POWER  
TO KNOW.

# **SAS<sup>®</sup> Data Surveyor for Clickstream Data 2.2 User's Guide**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2011. *SAS® Data Surveyor for Clickstream Data 2.2: User's Guide*. Cary, NC: SAS Institute Inc.

**SAS® Data Surveyor for Clickstream Data 2.2: User's Guide**

Copyright © 2011, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-60764-879-6 (electronic book)

All rights reserved. Produced in the United States of America.

**For a hardcopy book:** No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, or otherwise, without the prior written permission of the publisher, SAS Institute Inc.

**For a Web download or e-book:** Your use of this publication shall be governed by the terms established by the vendor at the time you acquire this publication.

**U.S. Government Restricted Rights Notice:** Use, duplication, or disclosure of this software and related documentation by the U.S. government is subject to the Agreement with SAS Institute and the restrictions set forth in FAR 52.227-19 Commercial Computer Software-Restricted Rights (June 1987).

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, February 2011

SAS® Publishing provides a complete selection of books and electronic products to help customers use SAS software to its fullest potential. For more information about our e-books, e-learning products, CDs, and hard-copy books, visit the SAS Publishing Web site at [support.sas.com/publishing](http://support.sas.com/publishing) or call 1-800-727-3228.

SAS® and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are registered trademarks or trademarks of their respective companies.

---

# Contents

<i>About This Book</i> . . . . .	<i>vii</i>
<i>What's New in SAS Data Surveyor for Clickstream Data 2.2</i> . . . . .	<i>ix</i>

## PART 1 Introduction to SAS Data Surveyor for Clickstream Data 1

<b>Chapter 1 • About SAS Data Surveyor for Clickstream Data</b> . . . . .	<b>3</b>
What is SAS Data Surveyor for Clickstream Data? . . . . .	3
Prerequisites . . . . .	4
Upgrade Considerations . . . . .	5
How to Get Help for SAS Data Surveyor for Clickstream Data . . . . .	7
<b>Chapter 2 • Components</b> . . . . .	<b>9</b>
SAS Page Tag Components . . . . .	9
Clickstream Jobs . . . . .	9
Clickstream Transformations . . . . .	10
Application Server Software . . . . .	12

## PART 2 Preparing a Data Source 13

<b>Chapter 3 • Data Source Preparation</b> . . . . .	<b>15</b>
Data Sources . . . . .	15
Recommended Data Source . . . . .	16
<b>Chapter 4 • Preparing Standard Web Log Data</b> . . . . .	<b>19</b>
Overview of Preparing Standard Web Log Data . . . . .	19
Best Practices for Preparing Standard Web Log Data . . . . .	19
<b>Chapter 5 • Preparing SAS Page Tag Data</b> . . . . .	<b>21</b>
Preparing SAS Page Tag Data . . . . .	21
Best Practices for Page Tagging Implementation . . . . .	22
Preparing the Clickstream Collection Server . . . . .	22
Inserting SAS Page Tag Code . . . . .	23
Inserting a Minimal SAS Page Tag . . . . .	23
Inserting a Full SAS Page Tag . . . . .	24
Customizing a Full SAS Page Tag . . . . .	26
Configuring SAS Real-Time Decision Manager Integration . . . . .	31

## PART 3 Processing Data with Clickstream Jobs 39

<b>Chapter 6 • Best Practices for Clickstream Jobs</b> . . . . .	<b>41</b>
Best Practices for Clickstream Jobs . . . . .	41

<b>Chapter 7 • Choosing a Starting Tutorial or Template Job</b>	<b>45</b>
Overview	45
Choosing a Starting Tutorial or Template Job	46
Copying the Folder Structure of a Clickstream Job	50
<b>Chapter 8 • Getting Started by Working with Tutorial Jobs</b>	<b>53</b>
Overview of Tutorial Jobs	53
About the Standard Web Log Tutorial Job	53
Stages in the Standard Web Log Tutorial Job	54
Running the Standard Web Log Tutorial Job	56
Running the SAS Page-Tag Tutorial Job	58
<b>Chapter 9 • Using Template Jobs</b>	<b>63</b>
About Template Jobs	63
Best Practices for Standard Template Jobs	66
Stages in Template Jobs	66
Running a Basic Template Job	74
About the Customer Intelligence Template Jobs	79
Collecting Campaign Information in a Customer Intelligence Job	79

## PART 4 Performing Clickstream Tasks 81

<b>Chapter 10 • Performing Common Tasks</b>	<b>83</b>
Common Clickstream Tasks	83
Selecting Files to Process	84
Using Character Decoding Options to Process Multi-Byte Data	84
Filtering Unwanted Data	86
Extracting Data from Clickstream Parameters	86
Tracking Internal Searches	88
Tracking Internal and External Referrers	90
Managing Non-Human Visitor Detection	90
Processing Campaign Data	91
Propagating Columns in Jobs that use the Loop Transformation	95
<b>Chapter 11 • Performing Page Tagging Tasks</b>	<b>97</b>
Configuring Link Tracking in Tagged Pages	97
Integrating SAS Real-Time Decision Manager	100

## PART 5 Clickstream Transformations 101

<b>Chapter 12 • Log Transformation</b>	<b>103</b>
About the Clickstream Log Transformation	103
Specifying the Path to the Log	104
Maintaining Log Types	105
Managing User Columns	106
Specifying Log Options	107
<b>Chapter 13 • Parse Transformation</b>	<b>109</b>
About the Clickstream Parse Transformation	109
Best Practices for the Clickstream Parse Transformation	111
Identifying Incoming Columns	112

Maintaining User Columns . . . . .	113
Applying Clickstream Parse Rules . . . . .	114
Managing the Visitor ID . . . . .	116
Managing Output Table Columns . . . . .	117
Specifying Parse Options . . . . .	117
<b>Chapter 14 • Sessionize Transformation . . . . .</b>	<b>119</b>
About the Clickstream Sessionize Transformation . . . . .	119
Best Practices for the Clickstream Sessionize Transformation . . . . .	122
Visitor ID Completion . . . . .	122
Spanning Web Logs . . . . .	123
Specifying Options for the Sessionize Transformation . . . . .	123
<b>Chapter 15 • Specialized Transformations . . . . .</b>	<b>127</b>
About the Specialized Transformations . . . . .	127
Directory Contents Transformation . . . . .	128
Clickstream Setup Transformation . . . . .	128
Clickstream Create Detail Transformation . . . . .	128
Clickstream Create Groups Transformation . . . . .	128
 PART 6 <b>Appendixes</b> 131	
<b>Appendix 1 • SAS Page Tag Predefined Data Elements Reference . . . . .</b>	<b>133</b>
<b>Appendix 2 • Clickstream Standard Columns Reference . . . . .</b>	<b>137</b>
<b>Appendix 3 • Server Software Reference . . . . .</b>	<b>145</b>



# About This Book

---

## Audience

The SAS Data Surveyor for Clickstream Data is designed for the following users:

- persons, such as data integrators, who are responsible for creating jobs that extract and transform clickstream data from Web logs and then loading the resulting data into a SAS table. Other applications, such as SAS Web Analytics, can then take the refined clickstream data and analyze it.
- persons who need to review the output of the clickstream jobs
- persons who would like to integrate SAS Real-Time Decision Manager capability into Web sites

You might be assigned to a specific role that determines which tasks you can perform. This documentation describes the tasks that can be performed with the SAS Data Surveyor for Clickstream Data. If you are not authorized to perform specific tasks, then your SAS Data Surveyor for Clickstream Data interface will not display those options.

Suggestions for using this document are as follows:

- For information about new features in this release, see [“What’s New in SAS Data Surveyor for Clickstream Data 2.2” on page ix](#).
- For an overview of the software, see [“About SAS Data Surveyor for Clickstream Data” on page 3](#) in Part 1: Introduction to SAS Data Surveyor for Clickstream Data.
- For information about how to select and prepare a data source for processing, see [“Data Sources” on page 15](#) in Part 2: Preparing a Data Source and [“Getting Started by Working with Tutorial Jobs” on page 53](#) in Part 3: Processing Data with Clickstream Jobs.
- For information about how to process a data source, see [“Best Practices for Clickstream Jobs” on page 41](#) in Part 3: Processing Data with Clickstream Jobs.
- For information about how to perform common tasks during processing, see [“Performing Common Tasks” on page 83](#) in Part 4: Performing Clickstream Tasks.
- For a detailed introduction to the main transformations that are used in clickstream jobs, see [“Log Transformation” on page 103](#), [“Parse Transformation” on page 109](#), and [“Sessionize Transformation” on page 119](#) in Part 6: Clickstream Transformations.
- For information about specialized clickstream processing, see [“Using Template Jobs” on page 63](#).





# What's New in SAS Data Surveyor for Clickstream Data 2.2

---

## Overview

SAS Data Surveyor for Clickstream Data 2.2 contains the following new and enhanced features:

- SAS Customer Intelligence Integration
- SAS Real-Time Decision Manager Integration
- Support for Multi-Byte Character Data
- Enhanced SAS Page Tagging Functionality
- Enhanced or New Clickstream Jobs
- Enhanced User Agent Processing
- Support for Tracking Internal Searches
- Enhanced Log Type Handling

---

## Customer Intelligence Integration

SAS Data Surveyor for Clickstream Data is part of the Customer Intelligence suite of solutions and integrates with the SAS Web Analytics, SAS Marketing Automation, SAS Digital Marketing, and SAS Real-Time Decision Manager products. This integration provides support for processing campaign information and preparing the information for analysis and reporting.

---

## SAS Real-Time Decision Manager Integration

Integration with SAS Real-Time Decision Manager allows for real-time campaign content to be presented to the Web site visitor based on information specific to the visitor's session. Any subsequent activity as a result of user actions taken upon the presented content will

be tracked. This can help with determining the success of campaigns and analyzing customers' responses to different types of content presented within a campaign.

---

## **Support for Multi-Byte Character Data**

New support for multi-byte character data provides the capability to process and display data containing multi-byte encoded characters. New default processing uses the UTF-8 character encoding.

---

## **Enhanced SAS Page Tagging Functionality**

Enhanced SAS Page Tag functionality includes the ability to track the following:

- User responses to web content dynamically driven by SAS Real-time Decision Manager
- Different loads of the same page by the same user
- HTML Tag, ID, and Name attributes
- Internal search result counts

---

## **Enhanced or New Clickstream Jobs**

Clickstream Jobs have been re-organized to provide both tutorial and template jobs. Tutorial jobs provide a way to learn the basics of clickstream processing, while template jobs use parallel processing to perform high-performance ETL for production

environments. In support of tighter page tag integration with SAS Real-Time Decision Manager, a new job has been added.

---

## **Support for Tracking Internal Searches**

New columns are provided for tracking searches performed on pages internal to the Web site. In addition, by using the SAS Page tag functionality, search result counts can be obtained.

---

## **Enhanced User Agent Processing**

User agent processing has been enhanced to identify a larger number of browsers, browser versions, and operating systems, while improving the processing performance.

---

## **Enhanced Log Type Handling**

You now have greater control over character positioning when reading records from a web log. This feature enhances the ability to properly read various log types.



## **Part 1**

---

# Introduction to SAS Data Surveyor for Clickstream Data

### *Chapter 1*

**About SAS Data Surveyor for Clickstream Data** ..... 3

### *Chapter 2*

**Components** ..... 9



## Chapter 1

# About SAS Data Surveyor for Clickstream Data

---

<b>What is SAS Data Surveyor for Clickstream Data? . . . . .</b>	<b>3</b>
<b>Prerequisites . . . . .</b>	<b>4</b>
<b>Upgrade Considerations . . . . .</b>	<b>5</b>
Overview . . . . .	5
Upgrading Existing Jobs to Use a SAS Unicode Application Server . . . . .	5
Column Width Changes . . . . .	7
<b>How to Get Help for SAS Data Surveyor for Clickstream Data . . . . .</b>	<b>7</b>
Overview . . . . .	7
Other Documentation . . . . .	8

---

## What is SAS Data Surveyor for Clickstream Data?

*Clickstream* is a term used to describe the data that is collected from users as they access Web pages through various electronic devices. Clickstream data includes the stream of user activity stored in a log. Clickstream data can be collected and stored in a variety of ways. The SAS Data Surveyor for Clickstream Data provides the capability to process this data into meaningful results.

The SAS Data Surveyor for Clickstream Data is a product that consists of several components. Together, these components enable you to create jobs that extract and transform clickstream data from a clickstream data source (such as a standard Web log or SAS page tag log). Then you can load the resulting data into a SAS table. Other applications, such as SAS Web Analytics, can then take the refined clickstream data and analyze it.

The SAS Data Surveyor for Clickstream Data consists of tutorial jobs, template jobs, clickstream transformations, and other components. These components are accessible from within SAS Data Integration Studio. The inputs to the jobs can be standard Web logs or enhanced logs that include SAS page tag clickstream data from tagged pages.

The SAS Data Surveyor for Clickstream Data enables you to do the following:

- use SAS page tagging to gather clickstream data that is not logged by a Web server, such as user interaction with pages retrieved from a browser cache instead of the Web server
- use tutorial jobs to learn how clickstream processing works
- use template jobs for common process flows that are used to cleanse and enrich clickstream data
- customize the template jobs for your own logs and outputs

- automate the extraction of useful information from large volumes of clickstream data
- integrate with SAS Real-Time Decision Manager to dynamically update web pages and track visitor activity

## Prerequisites

You must satisfy the following prerequisites in order to use the SAS Data Surveyor for Clickstream Data 2.2:

- Integration with any SAS Customer Intelligence products (such as SAS Web Analytics, SAS Digital Marketing, or SAS Real-Time Decision Manager) requires that those products are installed and their prerequisites satisfied.
- All prerequisites for SAS Data Integration Studio 4.21 must be satisfied.
- You must understand how to create jobs and manage process flows in SAS Data Integration Studio.
- An administrator must have installed the SAS clickstream components that are described in the following table.

The following Clickstream components can be installed from the SAS Deployment Wizard:

**Table 1.1** Clickstream Components in the SAS Deployment Wizard

Installation Location	Description	Where Components Are Installed
SERVER	<b>SAS Data Surveyor for Clickstream Data Server Components.</b> Consists of one or more Application Server contexts that include macros and other server components required to execute clickstream jobs (jobs that include Clickstream transformations). We recommend that you configure these servers to support Unicode processing. See <a href="#">“Using a Unicode SAS Application Server”</a> on page 41.	On all SAS 9.2 Application Servers that execute clickstream jobs.
CLIENT	<b>SAS Data Surveyor for Clickstream Data Plug-ins.</b> Includes transformations that are required to build clickstream jobs. For more information, see <a href="#">“Clickstream Transformations”</a> on page 10 in Part 5: Clickstream Transformations.	On all computers where you want to use SAS Data Integration Studio to build clickstream jobs.
METADATA SERVER	<b>SAS Data Surveyor for Clickstream Data Metadata:</b> Includes tutorial and template jobs and other components that are required to build clickstream jobs. For more information, see Part 3: Processing Data Using Template Jobs.	On the SAS Metadata Server.



Installation Location	Description	Where Components Are Installed
MID-TIER	<p><b>SAS Data Surveyor for Clickstream Data Mid-Tier.</b></p> <p>Updates an existing Apache Web server to become the clickstream collection server. Any pages that are instrumented with the SAS page tag will record Web site visitor activity to this server. For more information, see <a href="#">“Preparing the Clickstream Collection Server”</a> on page 22.</p>	On a computer where an Apache HTTP Server has already been installed.

---

## Upgrade Considerations

### Overview

This section contains information that should be reviewed when you are upgrading from a prior release of the SAS Data Surveyor for Clickstream Data.

### Upgrading Existing Jobs to Use a SAS Unicode Application Server

We recommend that you use a SAS Unicode server to process your data in order to support multi-byte character set data that might appear in Web data.

If you have existing jobs deployed that have not been running on a SAS Unicode Application Server, the SAS tables created as output would have been created using the encoding of the original SAS Application Server. To make the most efficient use of a SAS server, you should store the data in Unicode format with an encoding of UTF-8. By default, a file inherits the current session encoding when it is created. One of your first steps in converting an application to run with the SAS Unicode server is to convert the legacy data files.

It might be possible to run these jobs without converting the file encoding of the SAS tables. However, this practice could result in poor performance and a message similar to the following appearing in the SAS log:

*Note:* Data file PERMLIB.CH.DATA is in a format that is native to another host, or the file encoding does not match the session encoding. Cross Environment Data Access will be used, which might require additional CPU resources and might reduce performance.

A utility macro has been provided to assist in the migration of the contents of SAS Libraries within your job to your SAS Unicode Workspace Server. We recommend that you back up all locations before you proceed with any conversion in case a restore is necessary.

The SAS tables that need to be converted depend on your template job. They can be found in the following libraries:

- Permanent Library
- Output Library

- Additional Output Library
- Pre-assigned INLIB Library

The macro below should be submitted on your SAS Unicode Application Server. From within SAS Data Integration Studio, select your SAS Unicode Application Server as the current SAS Application Server. Then, submit the following macro with the **Code Editor** under the Tools menu:

```
%macro utf8conv(rootpath=);

%if %sysfunc(fileexist(&rootpath)) %then
%do;

libname root "&rootpath";

/* Grab list of tables in library. */
proc sql noprint;
select memname into :tables separated by ' '
from dictionary.members
where upcase(libname) eq 'ROOT' and
upcase(memtype) eq 'DATA';
quit;

%if &sqlobs gt 0 %then
%do;
/* Move tables to work */
proc datasets nowarn nolist;
copy out=work in=root move;
select &tables / mt=data;
quit;

/* Copy tables back to original location
with NOCLONE option, which will result
in the use of the current system encoding.
*/
proc datasets nowarn nolist;
copy out=root in=work noclone;
select &tables / mt=data;
quit;

/* Clean up work */
proc datasets lib=work nowarn nolist;
delete &tables / mt=data;
quit;

libname root clear;
%end;
%else %put No tables found in root location &rootpath
%end;
%else %put ERROR: Root path &rootpath not found.;

%mend;
```

The macro can then be executed against each SAS library that contains SAS tables that need to be converted. For example, to convert all of the SAS tables that exist in the default locations for the Standard Web Log Basic template job, you would submit the following:

```

%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop1\
Parse1));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop1\
Parse2));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop1\
Parse3));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop1\
Parse4));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Permlib1));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Permlib2));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Permlib3));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Permlib4));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Permlib5));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Sessionize1));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Sessionize2));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Sessionize3));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Sessionize4));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Loop2\
Sessionize5));
%utf8conv(rootpath=%str(C:\ClickstreamTemplates\2.2\StandardWebLogs\Basic\Output));

```

Once the data is converted, it can be updated only in a UTF-8 session. Non-UTF-8 sessions can read the data but not write to the data.

## Column Width Changes

The widths of many of the clickstream standard columns have been modified in order to match the widths used by SAS Web Analytics.

During an upgrade installation of SAS Data Surveyor for Clickstream Data, column widths in existing jobs are modified to match the new widths. This modification does not affect any deployed jobs that might be in production use. The new widths appear the next time an existing job is opened. Redeploying the job after opening it results in the new column widths being used.

For information about the column widths for this release, see [“Clickstream Standard Columns Reference” on page 137](#).

---

# How to Get Help for SAS Data Surveyor for Clickstream Data

## Overview

The following help documents are available to you:

- The latest version of this document: This document can be obtained at <http://support.sas.com/documentation/onlinedoc/surveyor>.
- *SAS Data Surveyor for Clickstream Data 2.2 User's Guide* (this document): You can find the *SAS® Data Surveyor for Clickstream Data 2.2 User's Guide* from the Start Menu under **Programs** ⇒ **SAS** ⇒ **SAS Data Surveyor for Clickstream Data 2.2 User's Guide**. You can also find it in the SAS Data Integration Studio custom tree. Follow the selection path **Products** ⇒ **SAS Data Surveyor for Clickstream Data 2.2** ⇒ **Documentation**. Right-click the user guide and click **Open** to access it on the Web.
- SAS Data Surveyor for Clickstream Data Page Tagging JavaScript Reference. This reference can be located at <http://support.sas.com/clk22>.
- Upgrade Information. For more information about SAS Data Surveyor for Clickstream 2.2, see [SAS Data Surveyor for Clickstream 2.2](#) in *Planning for Maintenance Releases and Product Upgrades for SAS 9.2*.
- Embedded Help Topics: You can access Help topics that describe the interface of the property windows for transformations and job templates that you use when you work with the Clickstream product. Many of the fields in the properties windows are documented with short textual descriptions. You can also press F1 to display contextual help for any part of the Clickstream interface.

## Other Documentation

Because the SAS Data Surveyor for Clickstream Data is used in the context of Data Integration Studio, you might need to consult the *SAS Data Integration Studio User's Guide*. When you integrate with other SAS solutions (for example, SAS Real-Time Decision Manager), consult the documentation for the respective solution by visiting <http://support.sas.com>.

## Chapter 2

# Components

---

<b>SAS Page Tag Components</b> . . . . .	<b>9</b>
<b>Clickstream Jobs</b> . . . . .	<b>9</b>
<b>Clickstream Transformations</b> . . . . .	<b>10</b>
<b>Application Server Software</b> . . . . .	<b>12</b>

---

## SAS Page Tag Components

The SAS Page Tag includes JavaScript page code, which runs in the user's browser. This page code collects data to a clickstream collection server, which records the log files to be processed.

JavaScript Page Code is used to refer to the code that is inserted into existing Web pages in order to collect enhanced data about activity on those pages. The following varieties are included in the SAS Data Surveyor for Clickstream 2.2:

- SAS Page Tag – The main JavaScript code that collects user activity on a page.
- RTDM Module – An optional JavaScript module that can be included when integrating with the SAS Real-Time Decision Manager product.

The clickstream collection server is an Apache HTTP Server configured by your administrator during deployment of SAS Data Surveyor for Clickstream Mid-Tier. This Web server collects data sent via the JavaScript Page Code.

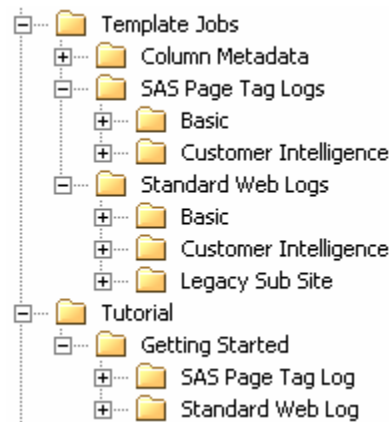
---

## Clickstream Jobs

The SAS Data Surveyor for Clickstream Data adds sample metadata for tutorials, template jobs, libraries, and tables to the tree view in SAS Data Integration Studio.

To see all of these objects together, display the Folders tree, expand the **Products** folder, the **SAS Data Surveyor for Clickstream Data** folder, and then the **2.2** folder, as shown in the following display;

**Display 2.1** Template Jobs Folder Structure



In addition to the Folders tree, clickstream jobs, libraries, and tables are also displayed under appropriate folders in the Inventory tree (jobs in the Job folder, and so on).

The following templates are installed with the SAS Data Surveyor for Clickstream Data:

- Tutorial jobs (found in the **Tutorial** folder) are provided for educational or testing purposes. These jobs enable you to become familiar with the basics of data processing using the Clickstream Transformations.
- Template jobs (found in the **Template Jobs** folder) are provided for use as the basis for production jobs used for ETL processing.

---

## Clickstream Transformations

SAS Data Surveyor for Clickstream Data adds a number of transformations to the Transformations tree in SAS Data Integration Studio. Most of these transformations are added to the **Clickstream Transformations** subfolder. The Directory Contents transformation is added to the **Access** subfolder.

The main clickstream transformations are Clickstream Log, Clickstream Parse, and Clickstream Sessionize. These transformations are used in every job, and they are responsible for the bulk of the ETL performed on clickstream data. For more detail about the functions of these transformations, see [“Log Transformation” on page 103](#), [“Parse Transformation” on page 109](#), and [“About the Clickstream Sessionize Transformation” on page 119](#).

The main transformations are described in the following table:

**Table 2.1** Clickstream Transformations

Name	Description	Inputs from and Outputs to
Clickstream Log transformation	<p>Reads data from a clickstream log. Identifies the type of log to be processed. Maps input fields from the log to the clickstream parse input columns as described in <a href="#">“Clickstream Parse Input Columns” on page 137</a>. Loads an output table with data from the log. For more information, see <a href="#">“Log Transformation” on page 103</a>.</p> <p>By default, this transformation will decode both URL encoded data and character encoded data.</p>	<p>From: Web log</p> <p>To: Log Output table</p>
Clickstream Parse transformation	<p>Reads the output from the Log transformation. Maps the clickstream parse input columns to output columns in a target table for continued processing. Filters unwanted data records from the target table, according to user-defined rules. Enables the definition of a cookie, a query string, or a referrer parameter to be parsed and stored as new data items in the target table. If possible, uniquely identifies the visitor who is associated with each data record and adds the visitor ID as a new data item in the target table. For more information, see <a href="#">“Parse Transformation” on page 109</a>.</p>	<p>From: Log Output table</p> <p>To: Parse Output table</p>
Clickstream Sessionize transformation	<p>Reads the output from the Parse transformation. Identifies user sessions. Performs additional visitor ID analysis. Identifies and manages non-human visitors (such as spiders). Manages sessions that span Web logs. For more information, see <a href="#">“Sessionize Transformation” on page 119</a>.</p>	<p>From: Parse Output table</p> <p>To: Sessionize Output table</p>

The following table describes the more specialized clickstream transformations in the order in which they are commonly used. Each of these transformations supports a special task in the template jobs that are installed with the SAS Data Surveyor for Clickstream Data.

**Table 2.2** Specialized Clickstream Transformations

Name	Description
Clickstream Setup transformation	<p>Generates the folder structure on the file system to hold the SAS logs and any generated data files. It also generates configuration data if necessary and test Web log data for the template jobs. Used in clickstream setup jobs.</p>

Name	Description
Directory Contents transformation	Generates a SAS data table that contains a numerical listing of the files found in a path or list of paths, and if selected, their subfolders. It is used in the Standard Web Logs Basic Template job as described in <a href="#">“Prepare Data and Parameter Values to Pass to Loop 1”</a> on page 66.
Clickstream Create Groups transformation	Combines the grouped output from several calls to the Clickstream Parse transformation into a set of output views, one per group. It is used in the Standard Web Logs Basic Template job as described in <a href="#">“Combine Groups”</a> on page 70.
Clickstream Create Detail transformation	Combines the output from multiple Clickstream Sessionize transformations and creates a single data table. It is used in the Standard Web Logs Basic Template job as described in <a href="#">“Create Detail and Generate Output”</a> on page 73.

For more detail about the functions of these transformations, see [“Specialized Transformations”](#) on page 127.

---

## Application Server Software

The back-end processing of data is supported by the SAS Data for Clickstream Data Server Software component. (This component was deployed by your administrator to one or more SAS Application Servers in your environment.) Once a job flow has been designed, the application server to which the job is submitted processes the data and creates the resulting output.



## **Part 2**

---

# Preparing a Data Source

<i>Chapter 3</i>	
<b>Data Source Preparation</b> .....	<i>15</i>
<i>Chapter 4</i>	
<b>Preparing Standard Web Log Data</b> .....	<i>19</i>
<i>Chapter 5</i>	
<b>Preparing SAS Page Tag Data</b> .....	<i>21</i>



## Chapter 3

# Data Source Preparation

---

<b>Data Sources</b> .....	<b>15</b>
Overview .....	15
Standard Web Logs .....	15
SAS Page Tag Logs .....	15
User-Defined Logs .....	16
<b>Recommended Data Source</b> .....	<b>16</b>

---

## Data Sources

### Overview

The SAS Data Surveyor for Clickstream Data is capable of processing data from standard Web logs, SAS page tag logs, and user-defined logs. Before you process data, you prepare the data source that you intend to process.

### Standard Web Logs

Standard Web logs are those logs that are collected by one of the following Web servers:

- Microsoft Internet Information Server
- Apache HTTP Server
- Sun iPlanet
- any other Web server that generates a Web log in one of the following file formats:  
Extended Log Format (ELF), Common Log Format (CLF), and Common Log Format Extended (CLFE)

### SAS Page Tag Logs

SAS page tag logs are those logs collected by a clickstream collection server (Apache HTTP server with the SAS Data Surveyor for Clickstream Data Mid-Tier Components installed). The content of these logs is generated by JavaScript page code that is inserted into the pages of interest on a Web site. SAS page tag log files are created in a custom log format supporting richer information than the information contained in a standard Web log.

When initially deploying the SAS Data Surveyor for Clickstream Data for a site, you will likely already have standard Web log data. As you incorporate the SAS page tag into your Web sites, you begin collecting richer data.

Identifying and understanding the data source that needs to be processed helps you to decide which tutorial or template jobs to select in the chapters to come. If you intend to collect SAS page tag data, then you need to perform extra tasks for inserting JavaScript page tag code into your Web pages. You also need to prepare one or more clickstream collection servers. (These tasks are described in Chapter 5: Preparing SAS Page Tag Data.) You do not have to perform these tasks if your data source is a standard Web log.

### **User-Defined Logs**

The Clickstream Log transformation supports the ability to create user-defined log types. This feature enables you to customize data source recognition and the processing for data sources that are not incorporated into the product by default.

---

## **Recommended Data Source**

The recommended data source is the SAS page tag log.

Although the SAS Data Surveyor for Clickstream Data processes standard Web server log files, these files are limited in the following ways:

- They provide a limited set of data.
- The data is captured only from the perspective of the Web server.
- The data includes every request to the Web server, even for files that are typically not of interest (such as image requests and spider or robot requests). This situation results in larger data volumes and a need to perform a great deal of filtering of the files.
- Some user actions are not captured. For example, browsers commonly cache pages. In that case, the use of the forward and back buttons in the browser does not result in a new request to the Web server. This processing results in user activity that is missed in the Web log.

These limitations of standard Web logs can be overcome with the use of a method of client-side (browser) data collection called page tagging. The page tagging method does not rely solely on the information that a Web server can gather. Instead, it uses the Web browser to gather data not normally logged by the Web server. The browser can gather this data because a small piece of code has been inserted into each page for which data is desired. This piece of code is known as a page tag. SAS provides a page tag solution with this product, which is referred to as the SAS page tag.

The SAS page tag runs inside of the user's Web browser when the user accesses a tagged page. The SAS page tag code has access to additional information from within the browser that is not normally available in a standard Web log. Once this data has been accessed in the browser, it is collected by sending it to a Web server. The Web server then stores in its Web log file only the requests for those pages that were tagged. When a Web server is used in this way (to collect clickstream data from tagged pages), it is referred to as a clickstream collection server. For a list of the data collected by the clickstream collection server, see [“Inserting SAS Page Tag Code” on page 23](#).

Working together, the SAS page tag code and one or more Web servers configured as clickstream collection servers provide a framework for client-side data collection. The actual data that is tracked is controlled with the SAS page tagging code that you insert. For

more information, see the SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference at <http://support.sas.com/c1k22>.



## Chapter 4

# Preparing Standard Web Log Data

---

<b>Overview of Preparing Standard Web Log Data</b> . . . . .	<b>19</b>
<b>Best Practices for Preparing Standard Web Log Data</b> . . . . .	<b>19</b>
Review Best Practices for Clickstream Jobs . . . . .	19
Use the Same Character Set on Web Pages . . . . .	19

---

## Overview of Preparing Standard Web Log Data

The SAS Data Surveyor for Clickstream Data is capable of processing the standard Web log types indicated in [“Data Sources” on page 15](#). Since these log files are generated by the Web server as the user activity was occurring on the Web site, little needs to be done to prepare these for processing.

Simply place the log files in a common location that can be accessed by the application server or servers that will process the logs. Note that all log data should be processed in chronological order.

---

## Best Practices for Preparing Standard Web Log Data

### ***Review Best Practices for Clickstream Jobs***

You should review the practices covered in [“Best Practices for Clickstream Jobs” on page 41](#) in addition to the following best practice.

### ***Use the Same Character Set on Web Pages***

When you process standard Web log data, it is the character set defined on the Web page that determines how the data sent to the server is encoded. Character set information is not included in a standard Web log record. Therefore, we have to assume that all records come from Web pages using the same character set. The recommended approach is to ensure that all Web pages whose accesses are logged to the same Web log use the UTF-8 character set. If they do not use UTF-8, then they should all use the same character set regardless and that value should be selected in the **Encoding of incoming data** option on the Clickstream Log transformation **Options** tab.

If the Web pages do not use the same encoding, then the **Decode incoming data?** option should be set to **No**.



## Chapter 5

# Preparing SAS Page Tag Data

---

<b>Preparing SAS Page Tag Data</b>	<b>21</b>
<b>Best Practices for Page Tagging Implementation</b>	<b>22</b>
Evaluating Security Issues for Form Capture	22
Securing Clickstream Collection Server Log Files	22
<b>Preparing the Clickstream Collection Server</b>	<b>22</b>
<b>Inserting SAS Page Tag Code</b>	<b>23</b>
<b>Inserting a Minimal SAS Page Tag</b>	<b>23</b>
Overview	23
Insert a Minimal SAS Page Tag	23
<b>Inserting a Full SAS Page Tag</b>	<b>24</b>
Overview	24
Insert a Full SAS Page Tag	24
<b>Customizing a Full SAS Page Tag</b>	<b>26</b>
Overview	26
Debug Mode	26
Page Load Tracking	26
Predefined Data Elements	27
User-Defined Data Elements	27
Meta Tag Tracking	28
Cookie Tracking	28
Link Tracking	28
Form Tracking	29
Rich Internet Application Tracking	30
<b>Configuring SAS Real-Time Decision Manager Integration</b>	<b>31</b>
Overview	31
Prerequisites	31
Configure the Web Pages	31
Defining the Call to the SAS Real-Time Decision Manager Web Service	32
Addressing Cross-Domain Issues	37

---

## Preparing SAS Page Tag Data

The SAS Data Surveyor for Clickstream Data is capable of processing data collected via the JavaScript SAS page tag code. SAS page tag log files are collected by a clickstream

collection server in response to JavaScript code inserted into each Web site page for which data collection is desired.

---

## Best Practices for Page Tagging Implementation

### *Evaluating Security Issues for Form Capture*

Form data collection is a powerful feature, but it is not enabled by default. When this capability is enabled, the data contained in the captured forms is stored in the clickstream collection server Web log as plain text. Access to this text presents a potential security issue. For example, a form capture option value of *1* collects all of the data contained in every form.

Therefore, you must ensure that the configuration settings for the form capture option result in the capture of the desired data only. You must also ensure that the clickstream collection server's Web log files are properly secured from unauthorized access. If you fail to properly configure this option, any sensitive data that a Web visitor submits in a form might be stored as raw text.

### *Securing Clickstream Collection Server Log Files*

As with any data in an organization, proper security measures should be put in place to protect sensitive information. Page tag data, by its very nature, contains information about user activity on a Web site. Depending on how the SAS page tag is configured for a site, sensitive information can be collected about user activity. All data collected by the SAS page tag is stored in the raw text Web log of the clickstream collection server. Appropriate measures should be taken to ensure that the Web log files for the clickstream collection server are properly secured from unauthorized access.

---

## Preparing the Clickstream Collection Server

You must install and configure the clickstream collection server that collects the output from the tags and writes that output to a SAS page tagging log.

*Note:* The Apache Software Foundation provides an open source Web server called Apache HTTP Server. This Web server is used as the clickstream collection server in a SAS page tagging environment. A working Apache HTTP Server (with or without SSL enabled) is a prerequisite to setting up this environment. Once this prerequisite is met, install the SAS Data Surveyor for Clickstream Data Mid-Tier software onto each Apache HTTP Server that you intend to use as a clickstream collection server.

#### **CAUTION:**

**You must configure and test the security of this server consistent with the standards of your organization both before and after you install the SAS Data Surveyor for Clickstream Data Mid-Tier components.** These precautions are necessary to protect the server from malicious attacks. In many cases, this server is exposed on the public Internet.

---

## Inserting SAS Page Tag Code

Once you have put one or more operational clickstream collection servers in place, the SAS page tag code needs to be inserted into the pages of interest for data collection. The data that is collected by this code is said to be tracked. The SAS Data Surveyor for Clickstream Data provides a robust page tag that enables the tracking of the following page components:

- page loads (even if forward and back buttons are used)
- link clicks
- form data elements such as POST data (turned off by default)
- cookie data elements such as name/value pairs and standard cookies
- meta tag data
- user-defined data elements such as name/value pairs
- rich Internet application events (such as Flash and AJAX)
- predefined data elements. For more information, see [“Predefined Data Elements” on page 27](#)

Most of the time, you insert a full SAS page tag that includes both the required lines and optional lines. This full tag enables you to customize your use of the clickstream collection server and collect additional types of data. For information about the various ways to insert the page code, see the sections in the remainder of this chapter.

---

## Inserting a Minimal SAS Page Tag

### Overview

You can insert a minimal SAS page tag into the pages of interest on your Web server. You can use this approach to minimize data elements collected, but a typical clickstream tagging implementation uses a full SAS page tag that yields more data because it can be customized. When you use a minimal tag configuration, the following default tracking settings are used:

- predefined data elements, which are collected on page load
- link clicks to all file types except the following: .htm, .html, .asp, .jsp, .asp, .cfm, .do, and .php
- all cookies
- all meta tags

Use the full tag configuration to modify default tracking configuration by using JavaScript API calls. For more information, see [“Inserting a Full SAS Page Tag” on page 24](#).

### Insert a Minimal SAS Page Tag

To tag a page, add the appropriate tag code to the end of the <BODY> section of each page of interest, right before the body close tag, </BODY>. The minimal code required to tag a page is as follows:

```
<script language="javascript" type="text/javascript"
  src="http://ccs.domain.com/sastag/SASTag.js"></script>
<script language="javascript" type="text/javascript">st_init();</script>
```

Before you insert the code into your pages, the protocol and domain **http://ccs.domain.com** must match the domain name of the clickstream collection server that contains the tag code. If you are collecting data over Secure Socket Layer (SSL), then the https prefix should be used instead of http.

---

## Inserting a Full SAS Page Tag

### Overview

Sometimes you want to process more than the default data elements provided by the minimal tag, and you want to be able to customize how the data is collected. In these cases, you can insert a full tag into the pages of interest on your Web server. This page tagging approach enables you to specify the data elements that you collect and customize the configuration for the tagging implementation. It also enables you to use debug mode during the initial integration and testing of the tag code that you are inserting into your Web pages. Most tagging implementations use full SAS page tags.

You can insert a full tag into the pages of interest on your Web server. You should understand that the minimal and full tag code yield the same data by default. Also, the minimal tag can be customized within the page if you use `st_pageCfg()` and `st_pageDats()` values. The full tag offers the following advantages:

- enables site-wide configuration values to be set by using `SASSiteCfg.js`
- enables the debug mode
- gathers a simple page load for browsers with JavaScript disabled

### Insert a Full SAS Page Tag

Insert the following code to the end of the `<BODY>` section of each page of interest, right before the body close tag, `</BODY>`. After the setup has been completed, data collection takes place in each tagged page by virtue of the call to `st_init()` that is made in the JavaScript.

```
01 <script language="javascript" type="text
  /javascript" src="http://ccs.domain.com/sastag/SASTag.js"></script>
02 <script language="javascript" type="text/javascript" src="http:
  //ccs.domain.com/sastag/SASSiteConfig.js"></script>
03 <script language="javascript" type="text/javascript">
04     function st_pageCfg {
05         // Place configuration values here
06     }
07     function st_pageDats {
08         // Place data values here
09     }
10 </script>
11 <script language="javascript" type="text/javascript" src="http:
  //ccs.domain.com/sastag/SASTagDebug.js"></script>
12 <script language="javascript" type="text/javascript">
  st_init();</script>
```

```
13 <noscript></noscript>
```

Before you insert the code into your pages, you must ensure that the protocol and domain **http://ccs.domain.com** match the domain name of the clickstream collection server that contains the tag code. If you are collecting data over Secure Socket Layer (SSL), then the https prefix should be used instead of http.

The following table provides a line-by-line explanation of the full page tag.

**Table 5.1** Line-By-Line Explanation of the Full SAS Page Tag

Line Number	Explanation
Line 1 (required)	Includes the SAS Tag code. This code includes and defines the <code>st_init()</code> function, as well as all of the data elements and default configuration settings for the tagging solution. After this line has been executed in the browser, <code>st_init()</code> is available to be called (line 12). Then the default tagging information is sent to the clickstream collection server.
Line 2 (optional)	Includes the default shared site configuration settings. This file is normally copied, renamed, and edited to set common site-specific configuration settings for data collection that applies across all pages into which it is included. The link in Line 2 then normally points to this copy.
Lines 3 to 10 (optional)	Enable page-specific configuration and data values to be set. Settings made here can override the product defaults and the site-wide settings and data values included in Line 2.
Line 11 (optional)	Useful to include when initially tagging pages to aid in testing. Inclusion of this line results in a pop-up debug window appearing as the data is being gathered. This feature provides more information about what is being collected. Note that you should be careful not to include this line in your production configuration or your users will also get this pop-up window. Also note that you must disable pop-up blocking software that prohibits this window from being displayed.
Line 12 (required)	Initializes the tagging code for the page. This line results in instrumentation of elements on the page and collection of data about the page load event.
Line 13 (optional)	Used when JavaScript is not enabled in the user's browser and the tag code cannot run. This line minimally makes an indication of this by requesting the tag image with a static set of information to indicate that JavaScript was not enabled. The page is tagged, but the information is not as rich as if JavaScript were enabled. This line is necessary only if you want to gather information about hits from users that have JavaScript disabled.

For information about the types of customizations that you can make to a full SAS page tag, see [“Customizing a Full SAS Page Tag” on page 26](#).

---

## Customizing a Full SAS Page Tag

### Overview

All data elements to be tracked are stored in the browser's memory before they are sent to the clickstream collection server. These data elements contain a key name, a value, and an enabled status. Only enabled keys have their values sent to the clickstream collection server. Tracking can be customized for each of the types of data that the page tag code is able to collect. This topic documents the procedure for the following elements:

- [“Debug Mode” on page 26](#)
- [“Page Load Tracking” on page 26](#)
- [“Predefined Data Elements” on page 27](#)
- [“User-Defined Data Elements” on page 27](#)
- [“Meta Tag Tracking” on page 28](#)
- [“Cookie Tracking” on page 28](#)
- [“Link Tracking” on page 28](#)
- [“Form Tracking” on page 29](#)
- [“Rich Internet Application Tracking” on page 30](#)

### Debug Mode

When you set up the initial tagging code, you can see the data that is sent to the clickstream collection server for a given page. This debug mode automatically scrolls to the top of the page. You can enable debug mode by simply inserting the line that includes `SASTagDebug.js` into the page tag code. For more information about this line, see [“Inserting a Full SAS Page Tag” on page 24](#).

When accessed, pages containing this line invoke a pop-up window that displays the configuration settings for the tag, data elements as they are being captured, and the exact data request sent to the clickstream collection server. The debug mode window continues to update as actions such as link clicks and form-submit-button clicks are performed on the page. Note that there can be only one debug mode window open for a given browser.

### Page Load Tracking

A page load event occurs anytime a page is loaded, refreshed, or returned to through the browser's navigational buttons (forward or back). The occurrence of any of these actions always results in data collection. The data that is collected can be configured.

Page load events are tracked using the load instance identifier (LID), which must first be generated and collected by the JavaScript page tag code. The LID element is added to the set of data elements that is sent to the clickstream collection server as a column in Clickstream Page Tag Data Elements User Columns table: It is set to a random number between 1 and 100,000 and collected on load, click, and submit events, as follows:

**Table 5.2** Page Load Instance Identifier Characteristics

Event	Description	Action	Sample Value
Load	Can include initial load, return to a page via back or forward buttons, or page refresh	Generates new LID	65324
Click	Includes a click action on a tracked item such as a link	Collects LID value of the page hosting the clicked item	65324 (assuming that the item clicked upon is on the same page that generated the LOAD event)
Submit	Includes submission of a form	Collects LID of the page hosting the submitted form	65324 (assuming that the form is on the same page that generated the LOAD event)

The LID element is added to the set of data elements that is sent to the clickstream collection server as a column in Clickstream Page Tag Data Elements User Columns table: You can find this table in the **Folders** pane using the following path: **Products** ⇒ **SAS Data Surveyor for Clickstream Data** ⇒ **2.2** ⇒ **Template Column Metadata** ⇒ **Clickstream Page Tag Data Elements User Columns**.

### Predefined Data Elements

Most of the predefined data elements have a value populated and are enabled for collection by default. Generally, predefined data element values should not be changed, unless an exception is documented. See “[SAS Page Tag Predefined Data Elements Reference](#)” on [page 133](#).

### User-Defined Data Elements

In cases where the predefined data elements do not provide enough information, user-defined data elements can be tracked. Code that tracks these data elements is added to either the `st_siteDats()` method in `SASSiteConfig.js`, or the `st_pageDats()` method in the full page tag code. Add the code to the former to track across multiple pages in your site or to the latter to track a data value for a specific page. For example, a content group value might be desired when several pages need to be classified as part of the same group of content. This value, if accessible from JavaScript, can easily be added to the set of data elements to track, as shown in the following code:

```
function st_siteDats()
{
  // Track a data value for all pages in this
  // site as "MarketingPages" since we
  // are dealing with our Marketing site.
  st_rq.dats.add("CONTENT_GROUP", "MarketingPages", true, 0x4
    /*capture on page load only*/);
}
```

This example collects a new data value for all pages that include `SASSiteConfig.js`. Using an externally included configuration file such as `SASSiteConfig.js` enables you to avoid editing the tagging code on each page to make global changes across multiple pages. If,

however, you would like to collect a data element for a specific page, such as the total on a shopping cart page, you can use the following page code:

```
function st_pageDats()
{
    // Track the shopping cart total for this page only
    st_rq.dats.add("CART_TOTAL",nTotal,true,0x1); /*capture on form submit only*/;
}
```

More information about the add() call can be found in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>. Look up “add” in the “Method Detail” section in the documentation for the ST\_Dats class.

## Meta Tag Tracking

By default, all meta tags on the page being accessed are tracked. Meta tag values are tracked by assigning an M\_ prefix to the name of the meta tag. For example, a meta tag value of CATEGORY with a value of BOOK is tracked as the name/value pair M\_CATEGORY=BOOK.

Meta tag tracking is configured by using the st\_cfg.cap[‘M’] array element. For example, you can turn off meta tag tracking with the following code in either the st\_siteCfg() or st\_pageCfg() methods: **st\_cfg.cap[‘M’] = “0”**; . You can also capture only the meta tag named Author with the following code: **st\_cfg.cap[‘M’] = “0:Author”**; . For details about configuring meta tracking, see the documentation for the st\_cfg.cap array element in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>.

## Cookie Tracking

By default, all cookies for the page being accessed are tracked. Cookie values are tracked by assigning a C\_ prefix to the name of the cookie. For example, a cookie named CART\_ID with a value of 32567 is tracked as the name/value pair C\_CART\_ID=32567.

Cookie tracking is configured by using the st\_cfg.cap[‘C’] array element. For example, you can turn off cookie capture with the following code in either st\_siteCfg() or st\_pageCfg(): **st\_cfg.cap[‘C’] = “0”**; . You can capture only chocolate, macadamia, and fudge cookies with the following code:

**st\_cfg.cap[‘C’] = “0:chocolate,macadamia,fudge”**; . For details about configuring cookies tracking, see the documentation for the st\_cfg.cap array element in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>.

## Link Tracking

A link on a page is tracked if the SAS page tag configuration results in the collection of data when the link is clicked. Links are tracked based on the file type of the target of the link. By default, links to all file types are instrumented with the exception of the following: htm, html, asp, aspx, cfm, do, and php. These link types are exceptions because these pages can be tagged. Therefore, their content can be directly tracked.

Link instrumentation is configured by using the st\_cfg.cap[‘L’] array element. The default setting of **st\_cfg.cap[‘L’] = “”**; tracks every link on a page. However, you can change the link configuration in either st\_siteCfg() or st\_pageCfg()). For example, you can enter **st\_cfg.cap[‘L’] = “0”**; to prevent the tracking of any links.



If you enable link tracking with `st_cfg.cap['L']="";`, you can also track specific types of links by configuring the `st_trk` element. See “[Configuring Link Tracking in Tagged Pages](#)” on page 97 for detailed information. For details about configuring link tracking, see the documentation for the `cap` array element in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>.

You can also add code to enable and disable the stop-and-re-click behavior that stops the user's initial click, collects data, and then re-clicks. Use the `st_trk()` method to determine whether an item is tracked and instrumented for data collection, as follows:

```
function st_trk(o)
{
  switch(o.nodeName.toLowerCase())
  {
    case 'a': // Link elements
      return true;
      break;
    default:
      return true;
  }
}
```

If you do enable tracking with `st_trk()`, you can use the `st_sar()` method to control how data is collected. This method enables you to use your own programming logic to control the stop-and-re-click behavior on and off, as follows:

```
function st_sar(o) {
  switch(o.nodeName.toLowerCase())
  {
    case 'a': // Link elements
      if ( o.href.indexOf('action=watch')>0 // MediaWiki watch/unwatch
          button handling
          || o.href.indexOf('action=unwatch')>0)
        return false;
      else
        return true;
      break;
    default:
      return true;
  }
}
```

For details about configuring stop and re-click behavior, see the documentation for `st_trk()` and `st_sar()` in the “*SASSiteConfig.js*” section in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>.

## Form Tracking

A form on a page is tracked if the SAS page tag configuration results in the collection of data when the user clicks a submit button on the form. Form element values are tracked by assigning an F prefix to the name of the form element. For example, a form element named `FIRST_NAME` with a value of John would be tracked as the name/value pair `F_FIRST_NAME=John`.

Form instrumentation is configured by using the `st_cfg.cap['F']` array element. For example, you can turn off tracking for every form on a page by changing the form configuration in either `st_siteCfg()` or `st_pageCfg()` as follows:

`st_cfg.cap['F']="0";` In addition, you can enter the following code:  
`st_cfg.cap['F']="1:1:formA,ccard,expdate:0:formB,fname,lname".`

This code tracks all of the forms on the page, but it (1) skips the elements named ccard and expdate in a form named formA and (2) captures only the elements named fname and lname in a form named formB. For details about configuring form tracking, see the documentation for the cap array element in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/clk22>.

*Note:* Form content (other than password fields) is not captured by default. If forms on your site do collect sensitive information, then this data is collected from the form. Then it is transmitted using the protocol of the collection server (http or https) and stored in the collection server's log file. Make sure that form tracking is configured with this in mind and store only what is appropriate. This sensitive information includes, but is not limited to, credit card numbers, bank account numbers, and personally identifiable information. Also, access to the clickstream collection server's Web log file should be restricted to authorized users only.

For more information about the security aspects of form data capture, see “Evaluating Security Issues for Form Capture” on page 22.

The tracking of form data requires the page that contains the forms to be tagged. Data collection is performed on the following form elements:

#### Text fields

```
<INPUT type="text">
```

#### Hidden fields

```
<INPUT type="hidden">
```

#### Password fields

```
<INPUT type="password">
```

Note that for password fields, the field name is collected, but the password value is not collected. A value of **x** is collected in place of the password. This setting is not configurable.

#### Text areas

```
<TEXTAREA>
```

#### Radio buttons

```
<INPUT type="radio">
```

#### Check boxes

```
<INPUT type="checkbox">
```

Check box data collection passes the value of each checked item, delimited by commas.

*Note:* As a general rule, if anything on a site requires the https protocol, then the site should use https for data collection. This precaution ensures that any sensitive information is transmitted securely.

## Rich Internet Application Tracking

A rich Internet application is an embedded object within a Web page that typically has its own self-contained functionality that is separate from the main HTML in the page. An example of this is an embedded Flash object.

Generally, any embedded object can be tracked if the object meets the following criteria:

- The object is programmable.
- Tracking code can be inserted at the point of interest.
- Calls to JavaScript can be made from the coding language of the object.

When these criteria are met, user actions within the rich Internet application can be tracked with the following JavaScript calls:

```

st_rq.dats.add("Name1","Value1",true,0x2 /* capture for click events only */);
st_rq.dats.add("Name2","Value2",true,0x2 /* capture for click events only */);
st_rq.dats.add("Name3","Value3",true,0x2 /* capture for click events only */);
st_rq.send(st_rq.RQST_CLK,"click");

```

These calls are documented in detail in the *SAS Data Surveyor for Clickstream Data 2.2 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/22M1/en/>. In particular, the call for add() is covered in the section for the ST\_Val() class and the eFlags field name.

For example, you can track user clicks from ActionScript within a Flash object by creating a trackEvent function in the coding language of the rich Internet application. The following code was created in Flash:

```

function trackEvent(event:MouseEvent):Void {
    ExternalInterface.call("st_rq.dats.add","EVNT",event.target);
    ExternalInterface.call("st_rq.send");
}

```

In this case, every mouse click passes an EVNT parameter and an indication of the user action that occurred.

---

## Configuring SAS Real-Time Decision Manager Integration

### Overview

See “Integrating SAS Real-Time Decision Manager” on page 100 for information about the added value that is provided by this functionality.

### Prerequisites

You must satisfy the following prerequisites:

- an Apache HTTP server that is configured as a clickstream collection server
- a Web page instrumented with a SAS page tag
- a Web page that includes the SASModRTDM.js file
- a Web page that defines a call to the Web service either inline or with an include
- a SAS Real-Time Decision Manager Web service that provides the appropriate response information for any requests

### Configure the Web Pages

You need to add the following elements to each Web page that you want use with RTDM:

- an anchor element
- a SAS page tag
- the RTDM module
- the code to call the SAS Real-Time Decision Manager

The anchor element enables SAS page tagging to tag a link for data collection. The RTDM module modifies the anchor element and its contents appropriately with values returned by the RTDM Web service. If the RTDM Web service fails, the anchor element is displayed without modification and acts as a default value. Here is an example of an anchor element:

```
<a href="http://yoursite.com" id="st_rtdm_id" alt="My Advert">10% off</a>
```

Two script tags are required to include the RTDM functionality on a Web page. One includes the SASModRTDM.js JavaScript file that is located on the clickstream collection server, as follows:

```
<script language="javascript"
type="text/javascript"src="http://ccs/sastag/SASModRTDM.js"></script>
```

The second tag should include the call to the RTDM Web service. This call can be declared directly on each page or in the SASSiteConfig.js file if the call is standard across your Web site. The call to the RTDM Web service in a separate site configuration JavaScript file is included in the following example:

```
<script language="javascript" type="text/javascript"
src="ccs/sastag/SASSiteConfig_site01.js"></script>
```

The final two script tags and one script tag enable the standard SAS page tag functionality, as shown in the following example:

```
<script language="javascript" type="text/javascript"
src="http://ccs/sastag/SASTag.js"></script>
<noscript>

</noscript>
<script language="javascript" type="text/javascript">st_init();</script>
```

## Defining the Call to the SAS Real-Time Decision Manager Web Service

An object is defined to make the call to the SAS Real-Time Decision Manager Web service. This object constructs the necessary request that is sent to the RTDM Web service. This request contains session information and then deconstructs the response to update the Web page content. The following example shows how an object might be defined:

```
function st_pageCfg()
{
    toRTDM = new ST_RTDM(
    {
        "event_name": "clickstream_fetch_dest",
        "url": "http://rtdm.yoursite.com/event",
        "target_element_id": "st_rtdm_id",                // default
        "image_link_element": "CONTENT_CODE",            // default
        "text_link_element": "CONTENT_TEXT",             // default
        "href_element": "DEST_URL",                     // default
        "request": [{"CUSTOMER_ID", "Float", global_cust_id},
                    ["PRODUCT_ID", "Int", global_product_id]],
        "capture": [{"RESPONSE_TRACKING_CODE", "RTDMRTC",
                    ["TREATMENT_TRACKING_CODE", "RTDMTTC"],
                    ["S1", "S1"], ["S2", "S2"],
                    ["S3", "S3"], ["S4", "S4"]]            // default
        }
    }
    );
}
```

*Note:* The default lines did not have to be specified in this example. They are included for completeness.

When inserted this code performs the following actions:

- calls the `clickstream_fetch_dest` operation from the Web service located at `http://rtdm.yoursite.com/event`
- internally constructs a SOAP request that contains the `CUSTOMER_ID` and `PRODUCT_ID` values. In this example the `global_cust_id` and `global_product_id` are JavaScript variables that are available from this page
- receives a response from the Web service, pulls out the image link information from the `CONTENT_CODE` element and the text link information from the `CONTENT_TEXT` element
- updates the anchor element in the current Web page that has the ID value of “`st_rtdm_id`” to display the image or text and appends the `RTDMRTC`, `RTDMTTC` and `S1` to `S4` query parameters to the link
- records the `RESPONSE_TRACKING_CODE` and `TREATMENT_TRACKING_CODE` `S1`, `S2`, `S3` and `S4` when the image or text is clicked by the user

The parameters for the call are explained in the following tables.

The following table covers the `event_name` parameter:

**Table 5.3** *event\_name*

<b>Parameter Name</b>	<code>event_name</code>
<b>Required?</b>	Yes
<b>Type</b>	String
<b>Default</b>	None
<b>Description</b>	The name of the operation to be invoked on the Web service
<b>Example</b>	“ <code>event_name</code> ”: “ <code>clickstream_fetch_dest</code> ”

The following table covers the `url` parameter:

**Table 5.4** *url*

<b>Parameter Name</b>	<code>url</code>
<b>Required?</b>	Yes
<b>Type</b>	String
<b>Default</b>	None
<b>Description</b>	The URL used to locate the Web service

<b>Example</b>	“url”: “http://comany.com/call”
----------------	---------------------------------

The following table covers the `target_element_id` parameter:

**Table 5.5** *target\_element\_id*

<b>Parameter Name</b>	target_element_id
<b>Required?</b>	No (default value searched for if no value specified)
<b>Type</b>	String
<b>Default</b>	st_rtdm_id
<b>Description</b>	Specifies the ID attribute of the anchor element in the Web page that is to be updated as a result of the values returned from the Web service
<b>Example</b>	“target_element_id”: “dynamic_ad”

The following table covers the `image_link_element` parameter:

**Table 5.6** *image\_link\_element*

<b>Parameter Name</b>	image_link_element
<b>Required?</b>	No (default value assumed if none is specified)
<b>Type</b>	String (case sensitive)
<b>Default</b>	CONTENT_CODE
<b>Description</b>	Identifies the XML element in the response from the Web service that contains the text to be displayed on the current Web page
<b>Example</b>	“image_link_element”: “CONTENT_CODE”

The following table covers the `text_link_element` parameter:

**Table 5.7** *text\_link\_element*

<b>Parameter Name</b>	text_link_element
<b>Required?</b>	No (default value assumed if none is specified)
<b>Type</b>	String (case sensitive)
<b>Default</b>	CONTENT_TEXT

<b>Description</b>	Identifies the XML element in the response from the Web service that contains the text to be displayed on the current Web page.  The behavior associated with this parameter depends on whether the value for the element specified in the <code>image_link_element</code> is set or blank. If the element that contains the <code>image_link_element</code> is empty, the text from the <code>text_link_element</code> is used to create a text advertisement. If the element that contains the <code>image_link_element</code> has a value, this <code>text_link_element</code> value is set as the ALT attribute.
<b>Example</b>	"text_link_element": "CONTENT_TEXT"

The following table covers the `href_element` parameter:

**Table 5.8** *href\_element*

<b>Parameter Name</b>	href_element
<b>Required?</b>	No (default value assumed if none is specified)
<b>Type</b>	String (case sensitive)
<b>Default</b>	DEST_URL
<b>Description</b>	Identifies the XML element in the response from the Web service that contains the value to be used for the HREF attribute on the anchor element. The anchor element is identified by the <code>target_element_id</code> parameter.
<b>Example</b>	"href_element": "DEST_URL"

The following table covers the `Input` parameter:

**Table 5.9** *Input*

<b>Parameter Name</b>	Input
<b>Required?</b>	Yes
<b>Type</b>	Array
<b>Default</b>	None
<b>Description</b>	Each element in this array defines a "Data" element in the generated SOAP request. Each element in the array should contain three values: the name of the XML element, its type, and finally its value.

<b>Example</b>	<pre> <input": "1050"}]="" "12237"},="" "float",="" "int",="" &gt;="" &lt;="" &lt;data="" &lt;float&gt;="" &lt;int&gt;="" &lt;val&gt;12272&lt;="" &lt;val&gt;6000&lt;="" <="" [{"customer_id",="" data&gt;="" float&gt;="" int&gt;="" name="PRODUCT_ID" pre="" q0:val&gt;="" {"product_id",=""> </input":></pre>
----------------	--

The following table covers the capture parameter:

**Table 5.10** *capture*

<b>Parameter Name</b>	capture
<b>Required?</b>	No (default value assumed if none specified)
<b>Type</b>	Array
<b>Default</b>	<pre> "capture": [{"RESPONSE_TRACKING_CODE", "RTDMRTC"}, ["TREATMENT_TRACKING_CODE", "RTDMTTC"]] </pre>
<b>Description</b>	<p>Each element in this array identifies the elements in the response XML whose values should be collected by the SAS Page Tagging Solution.</p> <p>Each element contains two values. The first identifies the response XML element that contains the value to be collected. The second identifies the query parameter name to be used to identify this data.</p>
<b>Example</b>	<pre> "capture": [{"TREATMENT_CODE", "RTDMTC"}, ["RESPONSE_TRACKING_CODE", "RTDMRTC"}, ["TREATMENT_TRACKING_CODE", "RTDMTTC"], ["S1", "S1"], ["S2", "S2"], ["S3", "S3"], ["S4", "S4"]] </pre> <p>Would result in the following parameters being records on the clickstream collection server:</p> <pre> 10.23.14.103 - - [02/Apr/2010:15:13:55 -0400] "GET /sastag/SASTag.gif?CS=ISO-8859-1\$VER=2.11\$EVT= click\$CID=ajax\$VID=12675421017167022847841310744\$URI =http://yoursites.com/\$REF=http://yoursite.com/pages /page01.html?uname=John &amp;product=bicycle\$TTL=http: //yoursite.com/content/saslogov1.gif\$PROT=http\$DOM =yoursite.com\$PORT=\$PLAT=Win32\$SINFO=1280x1024@24\$FL =1\$FLV=10.0%20r42\$CK=1\$JV=1\$JVV=1.6.0_18\$JS=1\$DT =4/2/2010\$TM=15:13:57.700\$RTDMTC=WA_Click01\$RTDMRTC =983100010501\$RTDMTTC=C80DBDCE9F9D813DE66D43559AD89E00\$S1 =12345.0\$S2=22345.0\$S3=32345.0\$S4=42345.0 </pre>



## Addressing Cross-Domain Issues

This functionality uses AJAX to communicate asynchronously with the RTDM Web service. AJAX does not allow requests to be made across different domains for security reasons. Therefore, a Web site that uses this SAS Real-Time Decision Manager integration must address these cross-domain issues. The example shown here uses the ProxyPass directive on the Apache server that serves the HTML pages to overcome the cross-domain issues. To enable this functionality, ensure that the following modules are loaded (either statically at compile time or dynamically with the LoadModule directive):

```
LoadModule proxy_module modules/mod_proxy.so
LoadModule proxy_http_module modules/mod_proxy_http.so
```

Then add the ProxyPass and ProxyPassReverse directives to your http.conf file, as shown in the following example:

```
ProxyPass /call http://specific_server_address/RTDM/Event
ProxyPassReverse /call http://specific_server_address/RTDM/Event
```

Please refer to the Apache HTTP Server Documentation for more information.

The URL (**`http://specific_server_address/RTDM/Event`**) specified in the ProxyPass and ProxyPassReverse statements points to your SAS Real-Time Decision Manager Web service. The “/call” value identifies the URL request that is to be forwarded to the Web service. In the ProxyPass statement above, the URL parameter of the RTDM call is defined as follows:

```
"url": "http://rtm.yoursite.com/call"
```

The Apache HTTP server recognizes the “/call” part of the URL request and forwards the request to the RTDM Web service.



## Part 3

---

# Processing Data with Clickstream Jobs

<i>Chapter 6</i>	
<b>Best Practices for Clickstream Jobs</b> .....	<a href="#">41</a>
<i>Chapter 7</i>	
<b>Choosing a Starting Tutorial or Template Job</b> .....	<a href="#">45</a>
<i>Chapter 8</i>	
<b>Getting Started by Working with Tutorial Jobs</b> .....	<a href="#">53</a>
<i>Chapter 9</i>	
<b>Using Template Jobs</b> .....	<a href="#">63</a>



## Chapter 6

# Best Practices for Clickstream Jobs

---

<b>Best Practices for Clickstream Jobs</b> . . . . .	<b>41</b>
Overview . . . . .	41
Order of Log Processing . . . . .	41
Using a Unicode SAS Application Server . . . . .	41
Data Source Encoding Considerations . . . . .	42
Backing Up Output Tables . . . . .	42
Resetting the CLICKRC Macro Variable . . . . .	43
Managing Permanent Library Tables . . . . .	43

---

## Best Practices for Clickstream Jobs

### *Overview*

The following best practices apply to all tutorial and template clickstream jobs.

### *Order of Log Processing*

All logs should be processed in chronological order. Clickstream Sessionize functionality requires this in order to avoid prematurely timing out visitor sessions.

### *Using a Unicode SAS Application Server*

SAS Unicode provides an environment for multiple language support, which includes support for the following tasks:

- handling different character encodings
- providing messages according to the locale and language of the user
- enabling a multi-lingual user interface

A SAS Unicode server is a particularly good option for Web data because most Web sites have a global presence. Therefore, they can receive request parameters (such as search terms) in multiple languages.

To use SAS Unicode server successfully, you must ensure that the SAS servers and all sessions that they invoke are SAS Unicode server sessions. There are multiple ways of setting up the environment to use SAS Unicode server. For the method that is best for your

environment, see <http://support.sas.com/resources/papers/92unicodesrvr.pdf>.

## Data Source Encoding Considerations

The character set encoding of the data source being processed should be understood when using the clickstream jobs. This is an important consideration because, by default, the clickstream jobs expect a UTF-8 character set encoding in the input raw log file.

If the data source encoding used in the incoming data differs from the **Encoding of the incoming data** setting in the Clickstream Log transformation within a job, the data is decoded incorrectly.

Therefore, you must verify the following before you process a data source:

- the value of the **Encoding of the incoming data** setting in the Clickstream Log transformation matches the encoding of the incoming data being processed
- the data records in the incoming raw log file have been encoded using the same encoding

If either of the above items is not met, then the **Decode input data?** option in the Clickstream Log transformation must be set to **No**. For example, the SAS page tag encodes all of the data that it collects in UTF-8. As a result, when the data source is a SAS page tag log, data will be decoded correctly.

On the other hand, standard Web logs contain data records that were encoded using the character encoding specified on each Web page by the Web site programmers. If the Web site follows a common standard encoding, the value of **Encoding of the incoming data** should be set to that encoding. If the Web site was not developed with a common encoding across Web pages, the **Decode input data?** setting in the Clickstream Log transformation should be set to **No** to maintain data integrity.

For more detail on the Clickstream Log decoding options and their usage, see the Help for the Input pane in the **Options** tab of the Clickstream Log transformation.

## Backing Up Output Tables

As in any production environment, it is good practice to backup your data.

By default, each execution of a SAS Data Integration Studio job overwrites the output tables created in the previous execution. If this is not what you want, then the output tables from each run should be retained.

*Note:* A clickstream job can produce large output tables. Make sure you monitor the disk space that is occupied by backups of these tables.

The following table lists the main output tables in a clickstream job and how these tables can be preserved after the job is executed.

**Table 6.1** Main Output Tables in Clickstream Jobs

Output Tables	How to Preserve Output Tables After the Job Is Executed
Data output table from a Sessionize transformation	<p>Back up the data output table for each Sessionize transformation.</p> <p>To identify the library and table to be backed up, display the properties window for the Sessionize output table. Click the <b>Physical Storage</b> tab. Note the name of the library and table. These data tables are typically named WEBLOG_DETAIL.</p>
Temporary work tables for parameters and rules that are output from the Parse transformation	<p>Redirect the temporary work tables for parameters and rules to a permanent library. Then back up this permanent library.</p> <p>To redirect the temporary work tables for parameters and rules, display the properties window for Clickstream Parse. Click the <b>Options</b> tab. In the <b>Tables</b> section, specify an <b>Additional output library</b>.</p>
Temporary work tables for spiders and sessions that are output from the Sessionize transformation	<p>Redirect the temporary work tables for spiders and sessions to a permanent library. Then back up this permanent library.</p> <p>To redirect the temporary work tables for spiders and sessions, display the properties window for Clickstream Sessionize. Click the <b>Options</b> tab. In the <b>Tables</b> section, specify an <b>Additional output library</b>.</p>
Permanent Library (Permlib) tables that are used to allow open sessions to be continued into the next execution of the Clickstream Sessionize transformation	See <a href="#">“Managing Permanent Library Tables”</a> on page 43 for more information about the usage of the Permanent Library tables.

## Resetting the CLICKRC Macro Variable

If there is a warning or error during the execution of a Clickstream transformation, then the return code variable CLICKRC might be set to a non-zero value for the transformation. This is done to prevent cascading failures in the rest of the job. To reset the CLICKRC value, do one of the following:

- Close and reopen the job. This creates a new SAS session on the application server.
- Open the properties window for the job, click the **Precode and Postcode** tab, and enter the following code in the **Precode** window: `%LET CLICKRC=0;`

*Note:* The CLICKRC macro variable is reset by default for all of the template jobs.

## Managing Permanent Library Tables

The Clickstream Sessionize transformation has the capability to retain data from sessions that were not closed when the last record of a log file was processed. These records are

then included when the next log file is processed. Clickstream Sessionize uses the location specified by the **Permanent library path** option setting of the Clickstream Sessionize transformation to store these records.

This behavior is desirable for production use. However, it is a best practice to process the same log file multiple times until the job is producing the desired output when you initially modify a clickstream job. In this case, the retained records are included every time the job is run, which results in duplicate records. To avoid this duplication, the Clickstream Sessionize **Delete PERMLIB tables** option is set to **Yes** by default.

When your clickstream job is producing the desired output from a single log file and you are ready to deploy the job for production use, ensure that this option is set to **No**. This practice results in unclosed sessions being retained between processing of each subsequent log file.

The tables located in the Permanent Library path location are the only tables generated by the Clickstream transformations that are used across ETL runs. Therefore, they require special consideration when a job needs to be rerun. You should ensure that you have a backup of the tables contained in the Permanent Library location before you run a job. If a failure occurs in the Clickstream Sessionize transformation, we recommended that you restore the tables in the permanent library location to their original state before you rerun the job.



## Chapter 7

# Choosing a Starting Tutorial or Template Job

---

<b>Overview</b> .....	<b>45</b>
<b>Choosing a Starting Tutorial or Template Job</b> .....	<b>46</b>
Overview .....	46
Standard Web Logs .....	47
SAS Page Tag Logs .....	48
Column Metadata .....	49
<b>Copying the Folder Structure of a Clickstream Job</b> .....	<b>50</b>

---

## Overview

If you have not yet prepared another data source, all tutorials and templates include a setup job that produces test data designed for use with the associated tutorial or template. Once the data source is ready for processing, the appropriate tutorial or template job must be chosen to process the data.

This chapter discusses how to select a starting tutorial or template job and how to make a working copy of the tutorial or template job structure. Tutorial jobs are intended for use for educational or testing purposes when you initially process small amounts of data. Template jobs are intended for use in the final ETL job flow that is eventually put into production.

## Choosing a Starting Tutorial or Template Job

### Overview

The tutorials and templates covered in the following table are available for your jobs:

**Table 7.1** Clickstream Templates

Name	Description
SAS Page Tag Log Tutorial ( <b>Tutorial</b> ⇒ <b>Getting Started</b> ⇒ <b>SAS Page Tag Log</b> )	Enables you to process a single clickstream log that includes SAS page tagging data.  Includes a setup job (clk_0010_tutorial_sastag_setup), the job template (clk_0020_tutorial_sastag_load_weblog_detail), and metadata objects for sample data under the Data Sources folder. For more information about this template, see <a href="#">“Running the SAS Page-Tag Tutorial Job” on page 58</a> SAS Page Tagging” on page 57.
Standard Web Log Tutorial ( <b>Tutorial</b> ⇒ <b>Getting Started</b> ⇒ <b>Standard Web Log</b> )	Enables you to process a single clickstream log as a tutorial when working with clickstream data.  Includes a setup job (clk_0010_tutorial_weblog_setup), the job template (clk_0020_tutorial_weblog_load_weblog_detail), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">“About the Standard Web Log Tutorial Job ” on page 53</a> .
Template Column Metadata ( <b>Column Metadata</b> )	Provides a repository of column definitions that are useful in clickstream jobs.  Includes the metadata for a number of tables and columns that are used in clickstream jobs. For detailed information about these tables, see <a href="#">“Clickstream Standard Columns Reference” on page 137</a> .
SAS Page-Tagging Basic Template ( <b>Template Jobs</b> ⇒ <b>SAS Page Tag Logs</b> ⇒ <b>Basic</b> )	Enables you to process clickstream logs that include page tagging data.  Includes a setup job (clk_0010_sastag_basic_setup), the job template (clk_0020_sastag_basic_load_weblog_detail), and metadata objects for sample data under the <b>Data Sources</b> folder.
SAS Page-Tagging Customer Intelligence Template ( <b>Template Jobs</b> ⇒ <b>SAS Page Tag Logs</b> ⇒ <b>Customer Intelligence</b> )	Enables you to process tagged Web logs that include information that allows for customer Web-based activity to be associated with the marketing campaign that originated the activity.  Includes a setup job (clk_0010_sastag_ci_setup), the job template (clk_0020_sastag_ci_load_weblog_detail), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">“Processing Campaign Data” on page 91</a> .

Name	Description
Standard Web Logs Basic Template ( <b>Template Jobs</b> ⇒ <b>Standard Web Logs</b> ⇒ <b>Basic</b> )	<p>Enables you to process clickstream logs that include data from standard Web logs from servers.</p> <p>Includes a setup job (clk_0010_weblog_basic_setup), the job template (clk_0020_weblog_basic_load_weblog_detail), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">“Using Template Jobs” on page 63</a>.</p>
Standard Web Logs Customer Intelligence Template ( <b>Template Jobs</b> ⇒ <b>Standard Web Logs</b> ⇒ <b>Customer Intelligence</b> )	<p>Enables you to process information that allows for customer Web-based activity to be associated with the marketing campaign that originated the activity.</p> <p>Includes a setup job (clk_0010_weblog_ci_setup), the job template (clk_0020_weblog_ci_load_weblog_detail), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">“Processing Campaign Data” on page 91</a>.</p>
Legacy Sub Site Template ( <b>Template Jobs</b> ⇒ <b>Standard Web Logs</b> ⇒ <b>Legacy Sub Site</b> )	<p>Enables you to process a single standard Web log that contains clickstream data for one or more subsites. The outputs include refined clickstream data for the entire site and for each subsite.</p> <p>Includes a setup job (clk_0010_weblog_sub_site_setup), the job template (clk_0020_weblog_sub_site_load_tables), and metadata objects for sample data under the <b>Data Sources</b> folder.</p> <p><i>Note:</i> This template is not available by default and is being deprecated. You must import it if you have users who have used it in a prior release and still need the functionality. This template will be omitted from future releases..</p>

In general, setup jobs generate the folder structure on the file system to hold the SAS logs and any generated data files. After you run the setup jobs, you can run the template jobs to verify that the template job is working properly.

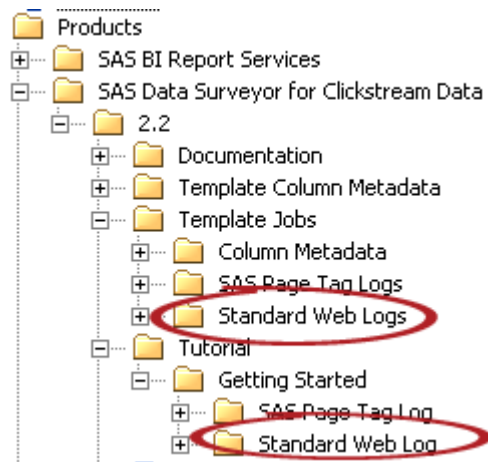
Several tutorial and template jobs are provided with the SAS Data Surveyor for Clickstream Data. These jobs are provided as the starting points for the most common ETL tasks. Choosing the correct template job from the beginning is crucial to ensuring the proper final output. These template jobs are divided into two classes. The first class is designed for processing standard Web logs and the second for processing SAS page tag logs.

## Standard Web Logs

You can find the template jobs that are designed for processing standard Web logs in the following location in SAS Data Integration Studio: **Folders** ⇒ **Products** ⇒ **SAS Data Surveyor for Clickstream Data** ⇒ **2.2** ⇒ **Template Jobs** ⇒ **Standard Web Logs**.

The templates and tutorials are shown in the following display:

**Display 7.1** Standard Web Logs in the Products Folder



When you expand the Standard Web Logs folder, you see the following types of Web log templates:

#### Basic

Designed for processing one or more standard Web logs from one or more standard Web servers. See [“Using Template Jobs” on page 63](#) for detailed information about this job.

#### Customer Intelligence

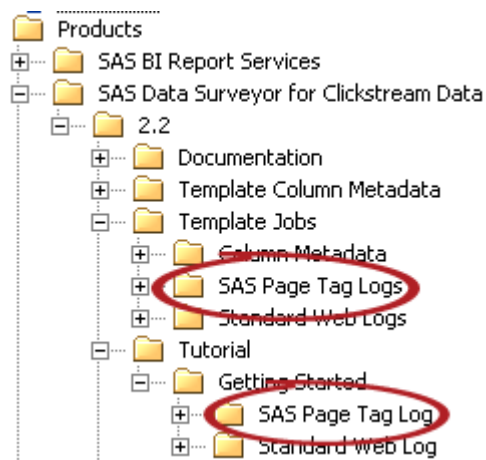
Designed to integrate data generated from customer campaigns when using the SAS Digital Marketing solution with standard Web logs. See [“Processing Campaign Data” on page 91](#) for detailed information about this job.

## SAS Page Tag Logs

The template jobs that are designed for processing SAS page tag logs are located at **Folders** ⇒ **Products** ⇒ **SAS Data Surveyor for Clickstream Data** ⇒ **2.2** ⇒ **Template Jobs** ⇒ **Page Tag Logs**.

The templates and tutorials are shown in the following display:

**Display 7.2** SAS Page Tag Logs in the Products Folder



When you expand the Page Tag Logs folder, you see the following types of Web log templates:

### Basic

Designed for processing one or more SAS page tag logs from one or more clickstream collection servers. See [“Running the SAS Page-Tag Tutorial Job” on page 58](#) for detailed information about this job.

### Customer Intelligence

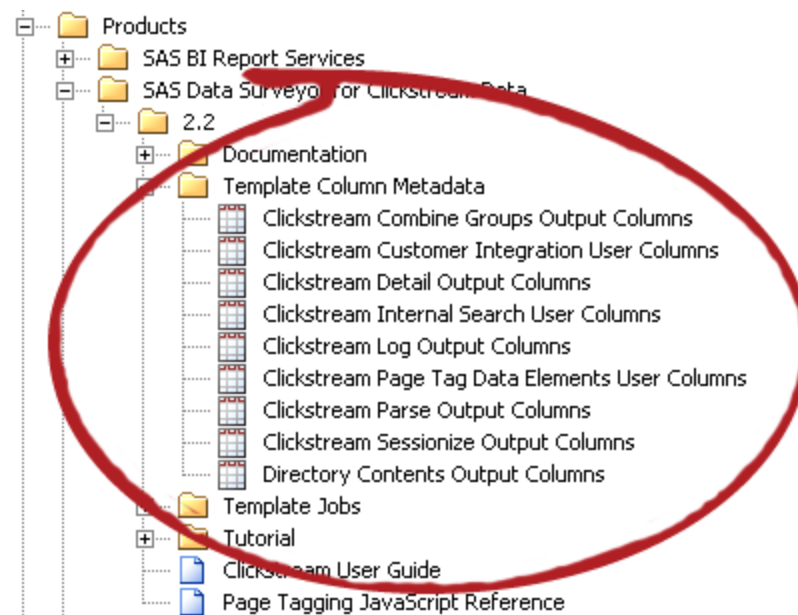
This template is designed to integrate data generated from customer campaigns when using the SAS Real-time Decision Manager and the SAS RTDM integration page tag module. If SAS page tags are used with the SAS Digital Marketing solution, then this job also supports integration of data generated from those campaigns. See [“Integrating SAS Real-Time Decision Manager” on page 100](#).

## Column Metadata

The Column Metadata folder does not contain template jobs. Instead, it contains table metadata with columns that are commonly used within the template jobs. The columns in these tables are useful when you work with copies of the template jobs because their metadata can be easily imported into jobs during development. The use of this column metadata enables you to avoid keying in commonly used column attributes. You can find this column metadata at the following location: **Folders ⇒ Products ⇒ SAS Data Surveyor for Clickstream Data ⇒ 2.2 ⇒ Template Jobs ⇒ Column Metadata**.

The column metadata folders are shown in the following display:

**Display 7.3** Column Metadata in the Products Folder



## Copying the Folder Structure of a Clickstream Job

You should copy the folder that contains the template before you modify any of the objects that it contains. When you use a copy of the template, you ensure that you keep the original template job and retain access to its default values.

All of the available job templates are located in the **Folders** pane. The directory paths are listed in the following table:

**Table 7.2** Locations of Clickstream Template Jobs

Template Job Name	Directory Path
Production Templates	
SAS Page Tag Basic Template	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Template Jobs\SAS Page Tag Logs \Basic
SAS Page Tag Customer Intelligence Template	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Template Jobs\SAS Page Tag Logs \Customer Intelligence
Standard Web Log-Basic	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Template Jobs\Standard Web Logs \Basic
Standard Web Log-Customer Intelligence	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Template Jobs\Standard Web Logs \Customer Intelligence
Tutorial Templates	
SAS Page Tag Log	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Tutorial\Getting Started\SAS Page Tag Log
Standard Web Log	Products\SAS Data Surveyor for Clickstream \2.2 Jobs\Tutorial\Getting Started\Standard Web Log

Perform the following steps to copy and prepare a Clickstream job template:

1. Right-click the folder that contains the template job and select **Copy**.
2. Right-click the folder where you want to paste the template and select **Paste Special** to access the Paste Special wizard. For example, you can paste the folder into the Shared Data folder if you want other users to have access to the new template. Note that you must have administrative privileges to perform this task.

*Note:* The decision to select **Paste Special** rather than **Paste** is very important. If you select **Paste**, then the paths in your copied job all point to the same paths used in

the original templates. **Paste Special** provides you the opportunity to change these paths while creating the copy.

Click **Next** to work through the pages in the wizard. You should leave all the objects selected in the Select Objects to Copy page. The SAS Application Servers page enables you to specify a default SAS Application server to use for the jobs that you are copying. The Directory Paths page enables you to change the directory paths that are used for objects such as SAS libraries. Click **Finish** when you complete the pages.

3. Rename (if desired) and expand the new folder that was just copied. Then, open the properties windows for the two jobs in the 2.2 Jobs folder and rename them. For example, you can gather Web log data that originates from a Web site designated as Site 1. In that case, you can rename the clk\_0010\_setup\_basic job to clk\_0010\_setup\_basic\_Site1 and the clk\_0020\_create\_output\_detail job to clk\_0020\_create\_output\_detail\_Site1.

*Note:* You might also need to rename the libraries for the single Web log, customer intelligence, and SAS page tag log job templates. Expand the Data Sources folder for the template and its subfolders to reveal the libraries used by the job. To distinguish these libraries from the original libraries used by the template job, you can rename these libraries to include the site name. For example, you can rename the Basic - Additional Output folder to Basic - Additional Output - Site1.

4. If you modified the Directory Paths when copying the templates, then open the renamed clk\_0010\_setup\_basic job and modify the Setup transformation properties. (Otherwise, proceed to step 6.) Then, on the **Options** tab, modify the values in the **Root Directory** and **Template Directory Name** fields to match the directory paths that you specified when creating the copy of this template. If you did not change the default values, then no changes should be required.
5. Run the renamed clk\_0010\_setup\_basic job. This job creates the necessary folders and sample data to support the renamed clk\_0020\_create\_output\_detail job.
6. Open the job properties window in the renamed clk\_0020\_create\_output\_detail job. Then, edit the EMAILADDRESS parameter in the **Parameters** tab.
  - First, select the EMAILADDRESS row in the table.
  - Second, click **Edit** to access the Edit Prompt window.
  - Third, click **Prompt Type and Value** and enter the e-mail address to use for any failure notification messages in the **Default value** field.
  - Fourth, click **OK** to exit the job properties window.
7. If you modified the directory paths when you copied the single log template, open the properties window for the Clickstream Log transformation and specify the appropriate value in the **File name** field on the **File Location** tab.





## Chapter 8

# Getting Started by Working with Tutorial Jobs

---

<b>Overview of Tutorial Jobs</b> .....	<b>53</b>
<b>About the Standard Web Log Tutorial Job</b> .....	<b>53</b>
<b>Stages in the Standard Web Log Tutorial Job</b> .....	<b>54</b>
Overview .....	54
Load and Prepare Clickstream Log Data .....	54
Parse Data .....	55
Create Sessions and Generate Output .....	55
<b>Running the Standard Web Log Tutorial Job</b> .....	<b>56</b>
Overview .....	56
Review and Prepare the Job .....	57
Run the Job and Examine the Output .....	58
<b>Running the SAS Page-Tag Tutorial Job</b> .....	<b>58</b>
Overview .....	58
Review and Prepare the Job .....	59
Run the Job and Examine the Output .....	59

---

## Overview of Tutorial Jobs

The tutorial jobs are designed to help familiarize you with the fundamental flow of processing in any clickstream job. Select a tutorial job based on the data source that you are working with. For more information, see [“Data Source Preparation” on page 15](#).

These tutorial jobs are located in the **Products** ⇒ **SAS Data Surveyor for Clickstream Data** ⇒ **2.2** ⇒ **Tutorial** folder in SAS Data Integration Studio. The tutorial jobs for both standard and SAS page tag Web logs are located in the Getting Started subfolder and include small amounts of test data.

---

## About the Standard Web Log Tutorial Job

The standard Web log tutorial template enables you to process only one clickstream log in a job. This template has been provided as a tutorial intended for learning about the structure of Clickstream templates and examining the properties of the Clickstream transformations. You can also use it to make a trial run on a single standard Web log to determine whether all of the data in the log can be read properly.

The standard Web log tutorial job template uses a Clickstream Log transformation to locate the log and a Clickstream Parse transformation to parse the log data into meaningful columns. It also uses a Clickstream Sessionize transformation to identify sessions within the data and generate output. The job also includes Checkpoint transformations to send error notifications when steps in the job fail.

You can use this tutorial to process a single standard Web log, regardless of the log file size. However, this circumstance is very rare in a production setting. In situations that require you to process multiple larger logs or enable you to split a single very large log, the standard Web logs basic template yields better performance because it uses parallel processing to process multiple logs at the same time. For more information, see [“About Template Jobs” on page 63](#).

## Stages in the Standard Web Log Tutorial Job

### Overview

The Single Log Template Job can be divided into the following stages:

- [“Load and Prepare Clickstream Log Data” on page 54](#)
- [“Parse Data” on page 55](#)
- [“Create Sessions and Generate Output” on page 55](#)

### Load and Prepare Clickstream Log Data

The Read Me First note in the job flow contains information needed for the initial setup and modification of this job. The only value described in this note is EMAILADDRESS, which supplies the e-mail address in the Checkpoint transformations in the template. This address is used for failure notification.

The first stage of the standard Web log tutorial job process uses a Clickstream Log transformation to locate the clickstream log data. Then it prepares a SAS data table or view that can be processed further.

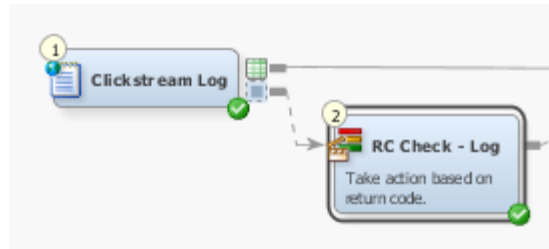
The transformations in this stage are described in the following table:

**Table 8.1** Load and Prepare Clickstream Log Data Transformations

Name	Description	Inputs from and Outputs to
Clickstream Log transformation	Extracts and decodes the raw Web log data with URL and character decoding. Creates a SAS data table or view.	From: Web log To: Clickstream Parse transformation; RC Check - Log transformation
RC Check - Log transformation	Evaluates the return code from Clickstream Log; sends e-mail to specified address if the log step fails.	From: Clickstream Log transformation To: Clickstream Parse transformation

The following display shows the portion of the tutorial job that runs this stage:

**Display 8.1** Load and Prepare Stage Process Flow



## Parse Data

The second stage of the standard Web log tutorial job process uses a Clickstream Parse transformation to parse the data and create meaningful columns.

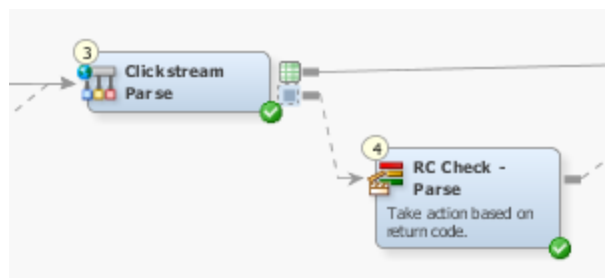
The transformations in this stage are described in the following table:

**Table 8.2** Parse Data Transformations

Name	Description	Inputs from and Outputs to
Clickstream Parse transformation	Parses the Web log data and transforms it into meaningful columns.	From: Clickstream Log transformation; RC Check - Log transformation To: RC Check - Parse transformation; Clickstream Sessionize transformation
RC Check - Parse transformation	Evaluates the return code from Clickstream Parse; sends e-mail to specified address if the log step fails.	From: Clickstream Parse transformation To: Clickstream Sessionize transformation

The following display shows the portion of the tutorial job that runs this stage:

**Display 8.2** Parse Data Stage Process Flow



## Create Sessions and Generate Output

The third stage of the standard Web log tutorial process uses a Clickstream Sessionize transformation to create sessions for the data and generate output in a detail table.

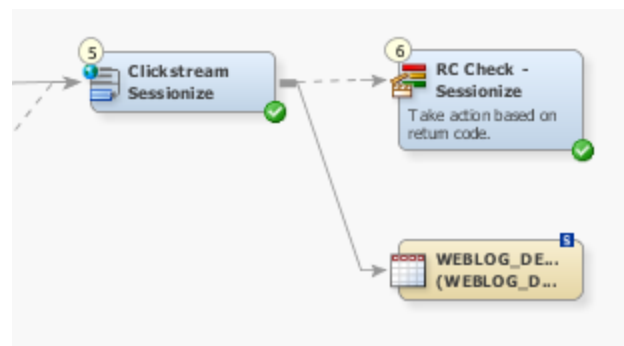
The transformations in this stage are described in the following table:

**Table 8.3** Create Sessions and Generate Output Table

Name	Description	Inputs from and Outputs to
Clickstream Sessionize transformation	Identifies sessions within the parsed data and creates a detail data table for further analysis.	From: Clickstream Parse transformation; RC Check - Parse transformation  To: RC Check - Sessionize transformation; WEBLOG_DETAIL table
RC Check - Sessionize transformation	Evaluates the return code from Clickstream Sessionize - ALL; sends e-mail to specified address if the sessionize step fails.	From: Clickstream Sessionize transformation  To: Next step in flow
WEBLOG_DETAIL table	Contains the output from the processed clickstream log file.	From: Clickstream Sessionize transformation  To: Another process or software component for further analysis (such as SAS Web Analytics)

The following display shows the portion of the job that runs this stage:

**Display 8.3** Sessions and Output Stage Process Flow



## Running the Standard Web Log Tutorial Job

### Overview

You can use the standard Web log tutorial job when you want to process a single clickstream log that was produced by a standard Web server. Note that the standard Web logs basic template generally provides better performance because it uses parallel processing. However, the standard Web log tutorial can help you learn about the structure of clickstream templates and the properties of clickstream transformations.

If you have not done so already, you should run a copy of the setup job for the tutorial Web log template, which is named `clk_0010_tutorial_weblog_setup`. The setup job creates test data and a folder called `ClickstreamTemplates` with subfolders to hold the output from the

standard Web log tutorial. When you actually process the data, you should run this copy of the standard Web log tutorial job, which is named `clk_0200_tutorial_weblog_load_weblog_detail`. For information about how to set up the copy of the required folder structure, see [“Copying the Folder Structure of a Clickstream Job” on page 50](#). By running a copy, you protect the original template.

Perform the following tasks to run a single log job:

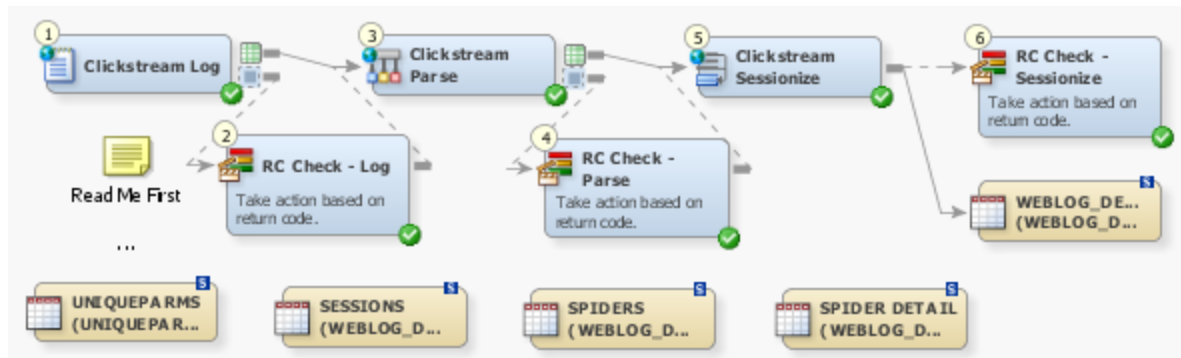
- [“Review and Prepare the Job” on page 57](#)
- [“Run the Job and Examine the Output” on page 58](#)

## Review and Prepare the Job

You can examine the standard Web log tutorial job on the **Diagram** tab of the SAS Data Integration Studio **Job Editor** before you run it. You can also specify the location of the clickstream log to process.

The following display shows a sample renamed tutorial job:

**Display 8.4** Copied Standard Web Log Tutorial Job



Note that the following tables are created as additional outputs in the Clickstream Parse and Clickstream Sessionize transformations:

- **UNIQUEPARAMS**: the unique parameters (as well as the type of parameter) found while processing the data
- **SESSIONS**: information about the sessions found while processing the data
- **SPIDERS**: information about the non-human visitor sessions found while processing the data
- **SPIDER DETAIL**: the detail activity of non-human visitor sessions identified in the **SPIDERS** table

These tables are created during the processing necessary to produce the final **WEBLOG\_DETAIL** output table. They are stored in the Additional Output library that is specified on the Clickstream Parse transformation or the Clickstream Sessionize transformation.

Perform the following steps to review and prepare the job:

1. Open the renamed copy of the standard Web log tutorial job.
2. Scroll through the job on the **Diagram** tab and review the following items:
  - the section that processes the source clickstream log
  - the section that parses the data into meaningful columns

- the section that creates sessions and generates an output table
3. Open the **File Location** tab in the properties window for the Clickstream Log transformation and review the file path to the clickstream log in the **File name** field. Specify another path if you need to process a different log. Click **OK** to close the properties window when you are finished.

*Note:* You can click the **Preview** button to view the first few lines of the file and confirm that you have selected a valid path.

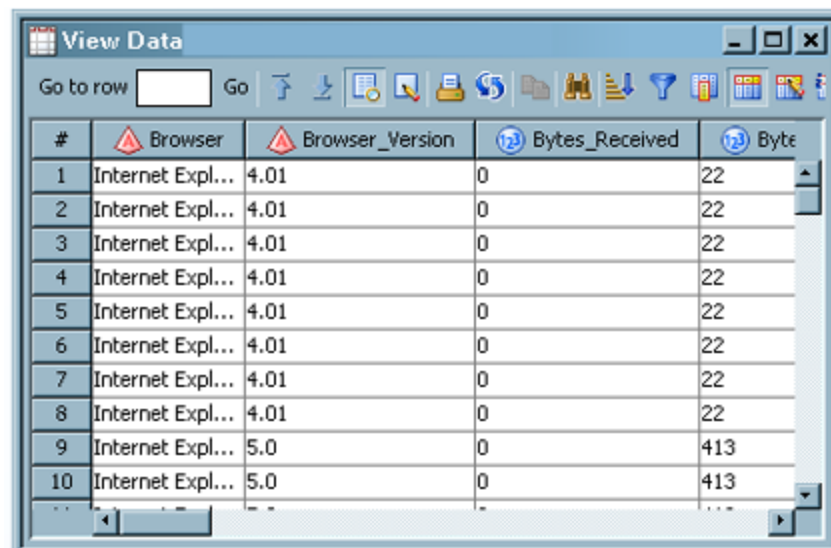
## Run the Job and Examine the Output

Perform the following steps to run a standard Web log tutorial job and examine its output:

1. Run the job.
2. If the job completes without error, right-click the OUTPUT\_DETAIL table at the end of the job and select **Open** in the pop-up menu.

The View Data window appears, as shown in the following display.

**Display 8.5** Standard Web Log Tutorial Job Output



#	Browser	Browser_Version	Bytes_Received	Byte
1	Internet Expl...	4.01	0	22
2	Internet Expl...	4.01	0	22
3	Internet Expl...	4.01	0	22
4	Internet Expl...	4.01	0	22
5	Internet Expl...	4.01	0	22
6	Internet Expl...	4.01	0	22
7	Internet Expl...	4.01	0	22
8	Internet Expl...	4.01	0	22
9	Internet Expl...	5.0	0	413
10	Internet Expl...	5.0	0	413

## Running the SAS Page-Tag Tutorial Job

### Overview

With SAS Data Integration Studio 4.2 and later, you can add notes to the job. A Read Me First note in the job flow informs the user to open the job properties window and edit the default value for the **Email Address for Checkpoint Notifications** parameter on the **Parameters** tab. The value that you set is used by all the Checkpoint transformations in this job. These Checkpoint transformations notify you when errors occur at strategic points in the job.

Perform the following tasks to run the SAS page tag default job:

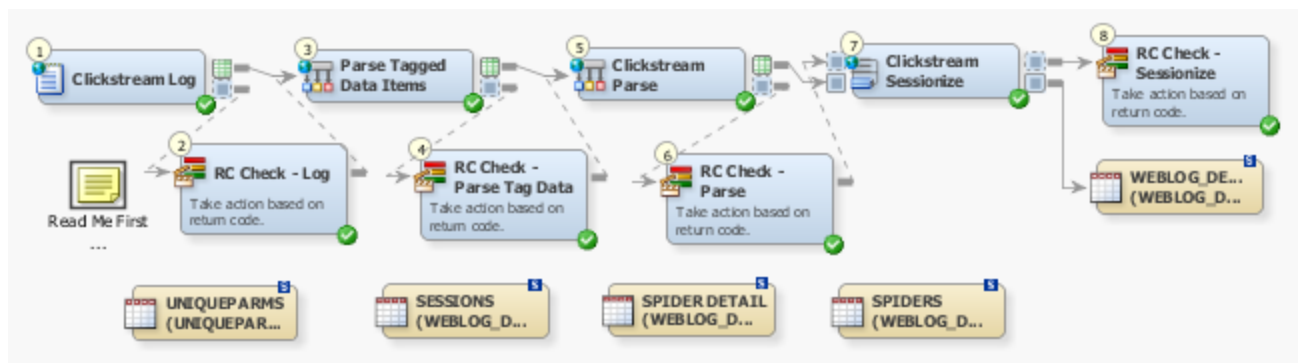
- “Review and Prepare the Job” on page 59
- “Run the Job and Examine the Output” on page 59

## Review and Prepare the Job

If you have not done so already, you should run a copy of the setup job for the SAS page tag tutorial, which is named `clk_0010_tutorial_sastag_setup`. When you actually process the data, you should copy and rename the page tag tutorial job before you run it. For information, see “[Copying the Folder Structure of a Clickstream Job](#)” on page 50. For example, you might run a job named `clk_0020_tutorial_sastag_load_weblog_detail_Site1`. Renaming a copy of the job ensures that you keep the original tutorial job and retain access to its default values.

The following display shows a sample renamed tutorial job.

**Display 8.6** Copied Page Tag Tutorial Job



Note that the following tables are created as additional outputs in the Clickstream Parse and Clickstream Sessionize transformations:

- **UNIQUEPARMS**: the unique parameters (as well as the type of parameter) found while processing the data
- **SESSIONS**: information about the sessions found while processing the data
- **SPIDERS**: information about the non-human visitor sessions found while processing the data
- **SPIDER DETAIL**: the detail activity of non-human visitor sessions identified in the SPIDERS table

These tables are created during the processing necessary to produce the final `WEBLOG_DETAIL` output table. They are stored in the Additional Output library that is specified on the Clickstream Parse transformation or the Clickstream Sessionize transformation.

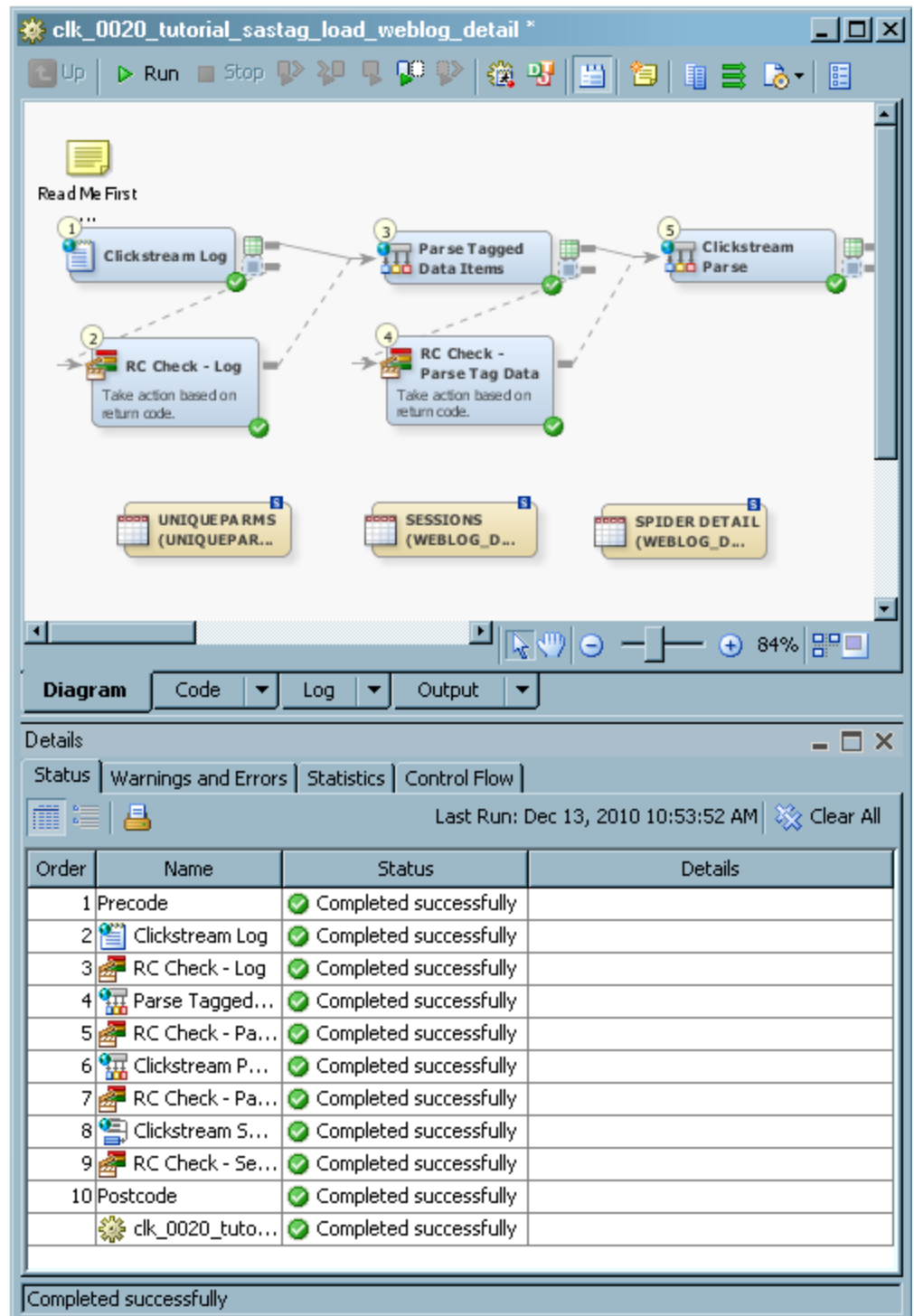
## Run the Job and Examine the Output

Perform the following steps to run the SAS page tag tutorial job and examine its output:

1. Open the job.

The following display shows a successfully completed job.

**Display 8.7** Completed Page Tag Tutorial Job



- If the job is completed without error, right-click the output table at the end of the job and select **Open** in the pop-up menu.



The View Data window appears, as shown in the following display.

**Display 8.8** Page Tagging Output

#	Collection_ID	Session_ID	Session_Closed	Entry_Point	
1	PTTEST	1267542101716...	0	1	0
2	PTTEST	1267542101716...	0	0	0
3	PTTEST	1267542101716...	0	0	0
4	PTTEST	1267542101716...	0	0	0
5	PTTEST	1267542101716...	0	0	2
6	PTTEST	1270237145183...	0	1	0
7	PTTEST	1270237145183...	0	0	2
8	PTTEST	1270239987265...	0	1	0
9	PTTEST	1270239987265...	0	0	0
10	PTTEST	1270239987265...	0	0	0
11	PTTEST	1270239987265...	0	0	2
12	PTTEST	1270240785738...	0	1	0
13	PTTEST	1270240785738...	0	0	0
14	PTTEST	1270240785738...	0	0	0
15	PTTEST	1270240785738...	0	0	2



## Chapter 9

# Using Template Jobs

---

<b>About Template Jobs</b> .....	<b>63</b>
<b>Best Practices for Standard Template Jobs</b> .....	<b>66</b>
Review Best Practices for Clickstream Jobs .....	66
<b>Stages in Template Jobs</b> .....	<b>66</b>
Overview .....	66
Prepare Data and Parameter Values to Pass to Loop 1 .....	66
Loop One: Recognize, Decode, Parse, and Group Data .....	68
Combine Groups .....	70
Loop Two: Sessionize .....	72
Create Detail and Generate Output .....	73
<b>Running a Basic Template Job</b> .....	<b>74</b>
Overview .....	74
Review and Prepare the Job .....	75
Run the Job and Examine the Output .....	76
Troubleshooting .....	78
<b>About the Customer Intelligence Template Jobs</b> .....	<b>79</b>
<b>Collecting Campaign Information in a Customer Intelligence Job</b> .....	<b>79</b>
Overview .....	79

---

## About Template Jobs

Template jobs are provided for processing both standard Web logs as well as SAS page tag logs. All template jobs are parameterized jobs that enable you to process multiple clickstream logs from the same or multiple servers. They also enable you to optimize processing time through the use of symmetric multi- processing using SAS MP Connect or grid computing. Finally, the templates manage outputs and resources to avoid contention.

*Note:* The SAS Data Surveyor for Clickstream included a template for processing subsites in clickstream data through version 2.1. As of version 2.2, this template is available only in legacy form, as a separate download.

These multiple template jobs use the same Clickstream Log, Clickstream Parse, and Clickstream Sessionize transformations that are used in the tutorial template jobs. The multiple clickstream log files are sent through a series of loops that are enclosed in the standard SAS Data Integration Studio Loop and Loop End transformations. In addition, several specialized transformations prepare the data and parameter values for the loops, group them to be sessionized, create detailed output, and generate an output table. The

Directory Contents transformation generates a list of raw Web logs to be passed into the first loop. Each iteration of the loop processes one Web log. Because they can handle more than one log, you will want to use them in your production environments.

The data is accessed from the raw Web logs by each parallel SAS session running in the first loop. Within the first loop, the Clickstream Log transformation reads a small number of the raw Web log records in order to determine the Web log type. Once the Web log type is determined, the transformation creates a SAS DATA step view that is used to read the raw Web log data. Still within the first loop, the Clickstream Parse transformation accesses the view built by the Clickstream Log transformation, and begins to process each incoming click observation as follows:

1. All AFTER INPUT rules are applied after an observation is initially read. Most filtering occurs here, where non-important data can be deleted very early in the process.
2. If the observation is not deleted, then the observation is parsed. This includes parsing of data such as the browser, browser version, platform, query parameters, referrer parameters, and cookies.
3. AFTER PARSE rules are then applied. Some filtering might occur here, if the decision to filter depends on parsed data. Otherwise, the filtering should be implemented using an AFTER INPUT rule.
4. Each observation is placed into an appropriate output group. The output group is decided using a grouping algorithm based on the Visitor ID or Client IP. (The algorithm also uses the User Agent when no Visitor ID value is supplied.) This practice ensures that all of the observations for a specific visitor session are stored in the same group. A list of group files created within each session is represented by L in [Figure 9.1 on page 65](#).

*Note:* You can configure the **Number of Groups** setting to optimize the job flow and support grid processing when identifying sessions. For example, entering the value **5** generates five groups. This setting enables you to execute up to five parallel sessionize loops.

The Clickstream Combine Groups generated transformation reads the group listing files and creates a SAS DATA step view that combines all the individual group files for a particular group. For example, the Group 1 data view accesses all of the group 1 data tables created during processing of the first loop. This transformation also creates a data table that is represented by G in [Figure 9.1 on page 65](#). This data table contains the list of data views that were created. This list is used to drive the second loop.

The second loop again takes advantage of symmetrical multi-processing to identify visitor sessions and to complete the visitor ID value from the start to the end of those sessions. This is accomplished using the Clickstream Sessionize transformation.

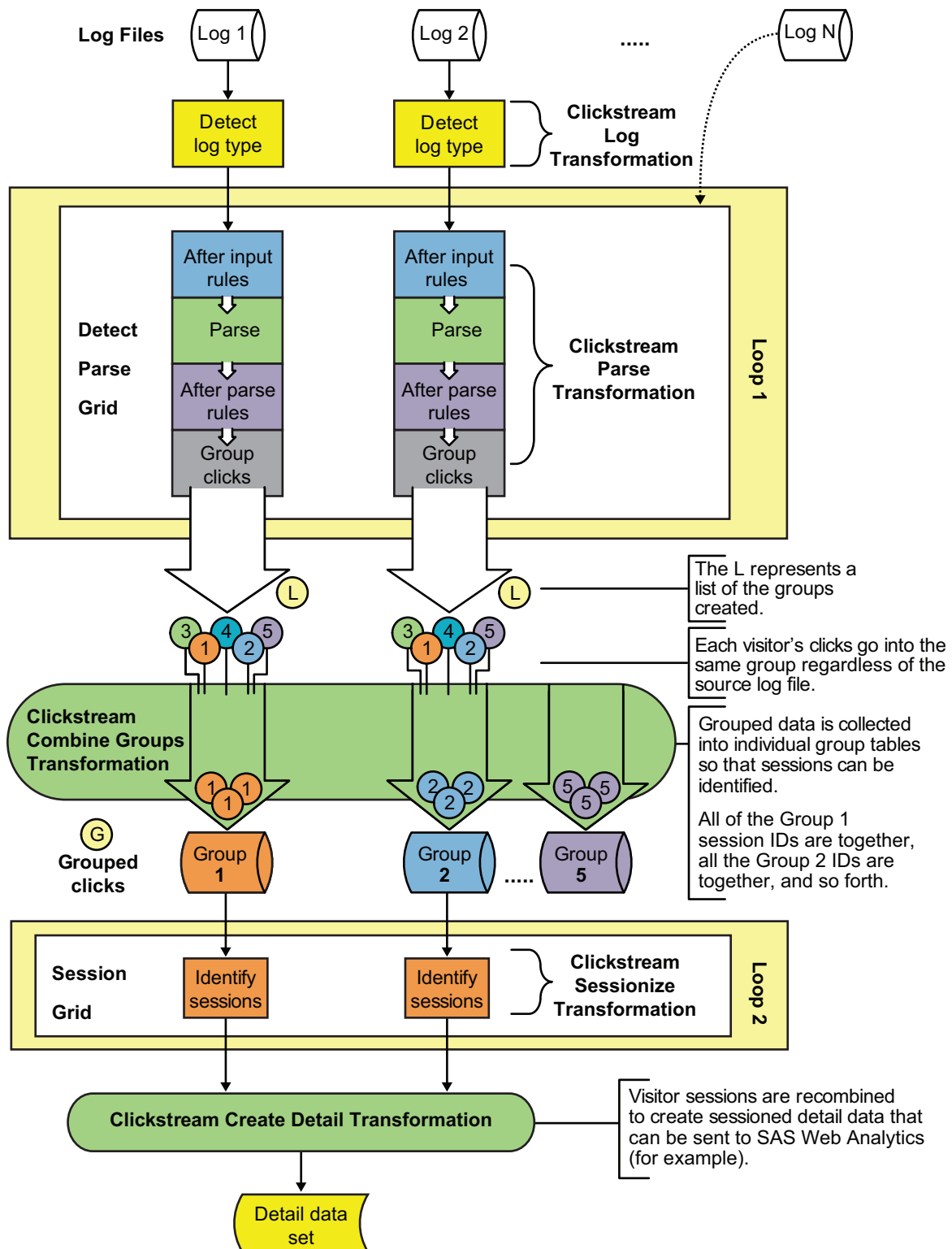
The completion of the visitor ID ensures that the visitor ID value that is assigned to users after they log on is present on every record of the session. This persistence holds even when the users browse the site for a period of time before logging in and after they log off. The visitor ID value is useful for connecting referring sites (purchased advertising, for example) to specific visitors and their final activity on the site (such as completing an online purchase).

Each parallel session reads observations from one of the group views created by the Clickstream Combine Groups transformation. Then it creates a single output data table in which sessions have been identified and visitor IDs have been completed.

After the second loop finishes, the Clickstream Create Detail transformation combines each output from the second loop to create the final composite detail data table.

The following figure illustrates the process flow for the standard Web log templates that are used to process multiple clickstream logs.

**Figure 9.1** Standard Clickstream Log Job Process Flow



When you process SAS page tag logs, Loop 1 contains two Clickstream Parse transformations. Sections of this figure are included in the descriptions of each stage of the template's processing.

*Note:* This description of multiple Web log processing is based on the template used to process a standard Web log. The process for the page-tagging template is slightly different, as noted in [“Stages in Template Jobs” on page 66](#).

---

## Best Practices for Standard Template Jobs

### *Review Best Practices for Clickstream Jobs*

In addition to examining the best practices listed here, you should also review the common best practices covered in [“Best Practices for Clickstream Jobs” on page 41](#).

---

## Stages in Template Jobs

### *Overview*

The template jobs can be divided into the following stages:

- [“Prepare Data and Parameter Values to Pass to Loop 1” on page 66](#)
- [“Loop One: Recognize, Decode, Parse, and Group Data” on page 68](#)
- [“Combine Groups” on page 70](#)
- [“Loop Two: Sessionize” on page 72](#)
- [“Create Detail and Generate Output” on page 73](#)

*Note:* This topic describes the stages in the standard Web log basic template job. The page-tagging version of the template contains the Parse Tagged Data Items transformation in the Loop One: Recognize, Parse, Group Data stage of the job. This transformation is a renamed Clickstream Parse transformation.

### *Prepare Data and Parameter Values to Pass to Loop 1*

The Read Me First note in the job flow contains information that is necessary for the initial setup and modification of this job. You might need to edit the following values in the **Parameters** tab for the job:

#### EMAILADDRESS

supplies the e-mail address in the Checkpoint transformations in the template. This address is used for failure notification.

#### NUMPARSEPATHS

determines the number of folders that are created for holding output for the parallel executions of the Clickstream Parse transformation in the first loop. Set this value to match the default value that was used in the setup job. If that value was changed in the setup job, then it should also be updated here.

## NUMGROUPS

determines how many groups of data are created by the Clickstream Parse transformation during the first loop. Therefore, it also determines the maximum number of parallel executions for Clickstream Sessionize during the second loop. Set this parameter value to match the default value that was used in the setup job. If that value was changed in the setup job, then it should also be updated here.

The first stage of the standard Web log basic template process locates the data and parses it.

The transformations and tables in this stage are described in the following table:

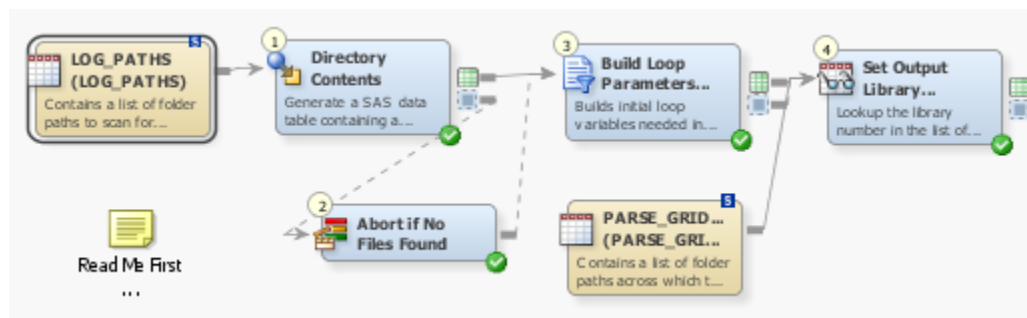
**Table 9.1** Locate and Parse Transformations and Tables

Name	Description	Inputs from and Outputs to
LOG_PATHS table	Contains a list of folder paths to scan for clickstream logs.	From: None To: Directory Contents transformation
Directory Contents transformation	Generates a data table that contains a list of the files found in the directories that are listed in the LOG_PATHS data table. The output table contains the following columns: <ul style="list-style-type: none"> <li>FILENUM: a unique sequence number related to that file (such as 1,2,3,4)</li> <li>FILENAME: the name of the file</li> <li>FULLNAME: a combination of path and filename</li> </ul>	From: LOG_PATHS table To: Build Loop Parameters (reused SAS Extract) transformation
Abort If No Files Found	Evaluates the return code from Directory Contents; stops the run if no files are found.	From: Directory Contents transformation To: Build Loop Parameters (reused SAS Extract) transformation
Build Loop Parameters (reused SAS Extract) transformation	Passes through the columns from the Directory Contents transformation and creates two additional columns. LIBRARYNUMBER is a number from 1 to $n$ where $n$ is the number of output locations that have been defined on the file system for the first loop (the Clickstream Parse transformation). This column's value is used to ensure that when running in parallel, the output from the jobs is spread across the different folders. PARSEOUTMEMBER uses the incoming FILENUM value to create a unique suffix for the parse output tables. This ensures that when two streams use the same folder, the output from one does not overwrite the output from the other.	From: Directory Contents transformation To: Set Output Library Locations (reused Lookup) transformation

Name	Description	Inputs from and Outputs to
PARSE_GRID_PATHS table	Contains a list of paths to folders where the outputs from multiple Clickstream Parse transformation calls are distributed. The paths specified in this table are accessed simultaneously by parallel processes. To optimize performance, specify paths that reside on different physical disks or network locations.	From: None  To: Set Output Library (reused SAS Extract) transformation
Set Output Library (reused SAS Extract) transformation	Uses the output library locations that are listed in the PARSE_GRID_PATHS configuration table. This transformation uses the LIBRARYNUMBER column to associate that log file with an output location (PARMLIBPATH) and an output LIBNAME (PARMLIBNAME). These values provide a different input file and output library for each iteration of the loop that follows.	From: Build Loop Parameters (reused SAS Extract) transformation, and PARSE_GRID_PATHS table  To: Loop 1 (Recognize and Parse) transformation

The following display shows the locate and parse data stage of the template job.

**Display 9.1** Locate and Parse Process Flow



### Loop One: Recognize, Decode, Parse, and Group Data

The second stage contains the first loop job. The transformations in the first loop job represent the subjob, which is the job that is run in parallel. For a standard Web log, each stream consists of a Clickstream Log transformation, a Clickstream Parse transformation, and two checkpoints, which are created by renaming the Return Code transformation and which enable you to configure how errors are processed.



The transformations in this stage are described in the following table:

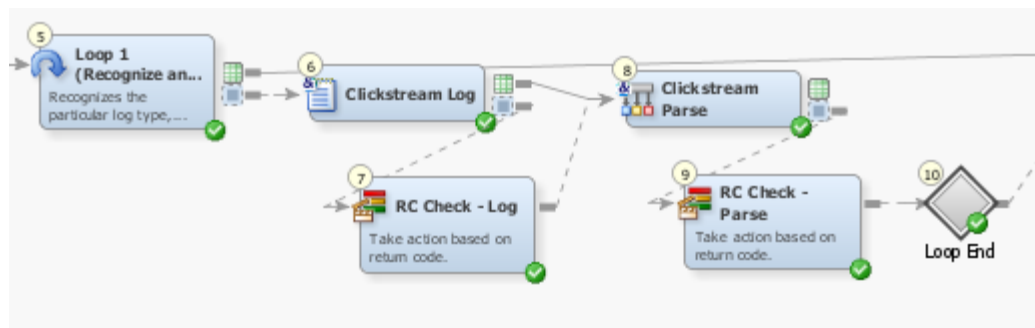
**Table 9.2** Loop One Transformations

Name	Description	Inputs from and Outputs to
Loop 1 (Recognize and Parse) transformation	<p>Passes the appropriate parameters through to the job flows that are executed in parallel. Each parallel stream should have the following parameters set:</p> <ul style="list-style-type: none"> <li>INPUTFILE is supplied by the FULLNAME source column</li> <li>OUTLIBPATH is supplied by the PARMLIBPATH source column</li> <li>INFILENUM is supplied by the FILENUM source column</li> </ul>	<p>From: Set Output Library (reused SAS Extract) transformation</p> <p>To: Clickstream Log transformation</p> <p>To: Filter - Only properly parsed logs (SAS Extract)</p>
Clickstream Log transformation	<p>Extracts and decodes (URL and character) data from a single log for each pass through the loop; determines the raw Web log type and creates a SAS DATA step view that is used to read the raw data.</p>	<p>From: Loop 1 (Recognize and Parse) transformation</p> <p>To: RC Check - Log transformation</p> <p>To: Clickstream Parse transformation</p>
RC Check - Log transformation	<p>Evaluates the return code from Clickstream Log; sends e-mail to specified address if the log step fails.</p>	<p>From: Clickstream Log transformation</p> <p>To: Clickstream Parse transformation</p>
Clickstream Parse transformation	<p>Parses this data and generates <math>n</math> output tables, where <math>n</math> is the number of groups expected by the Sessionize loop (the second loop).</p> <p>For customer intelligence template jobs, this step also identifies the campaign and customer who clicked on a specific treatment.</p> <p>Campaign information is denoted by these columns:</p> <ul style="list-style-type: none"> <li>EntrySource: ID of the entity that originated access to the landing page</li> <li>EntryActionID: ID that represents the Entry Source</li> <li>S1 through S4 - identifies the subject of an Entry Action either alone or with other Subject ID parameters</li> </ul> <p>Clickstream Parse populates EntrySource with a value of "SDM" if there is a value in the EntryActionID and S1 columns.</p>	<p>From: RC Check - Log transformation</p> <p>To: RC Check - Parse transformation</p>
RC Check - Parse transformation	<p>Evaluates the return code from Clickstream Parse; sends e-mail to specified address if the parse step fails.</p>	<p>From: Clickstream Parse transformation</p> <p>To: Loop End transformation</p>

Name	Description	Inputs from and Outputs to
Loop End transformation	Ends loop processing; returns to beginning of loop	From: RC Check - Parse transformation  To: Filter - Only properly parsed logs (reused SAS Extract) transformation

The following display shows the first loop stage of the template job.

**Display 9.2** Loop 1 Process Flow for Standard Web Logs



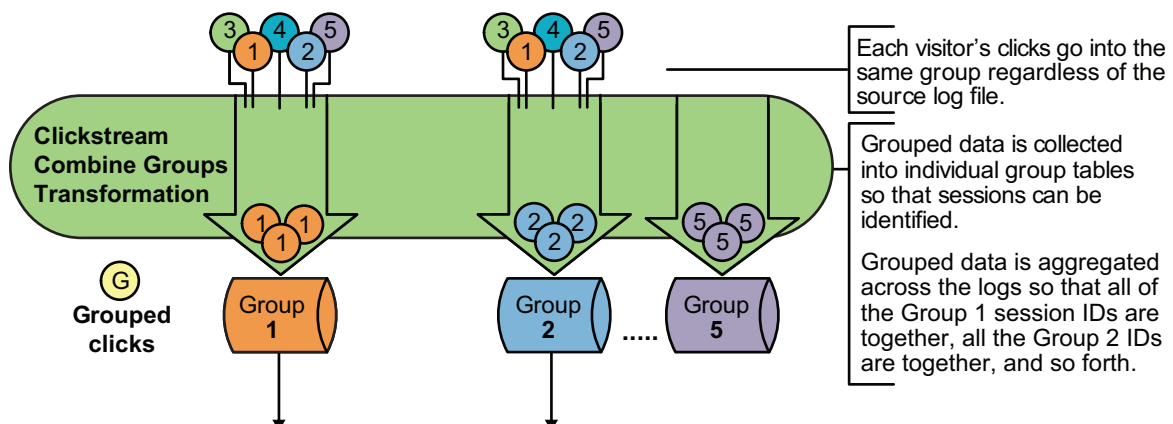
When you process SAS page tag logs, an additional Clickstream Parse transformation (and associated RC Check) are inserted in order to process the data elements collected by the SAS page tag (after the RC Check-Log transformation and before the Clickstream Parse transformation). This additional Clickstream Parse transformation is named Parse Tagged Data Items. For a partial list of the data elements processed by this additional transformation, see [“SAS Page Tag Predefined Data Elements Reference” on page 133](#).

## Combine Groups

The third stage prepares the groups used in the sessionizing process in the second loop. This stage contains transformations that filter for properly parsed logs, create groups, build loop parameters, and prepare paths and output locations for the upcoming loop.

The following figure illustrates this stage of the process:

**Figure 9.2** Combine Groups



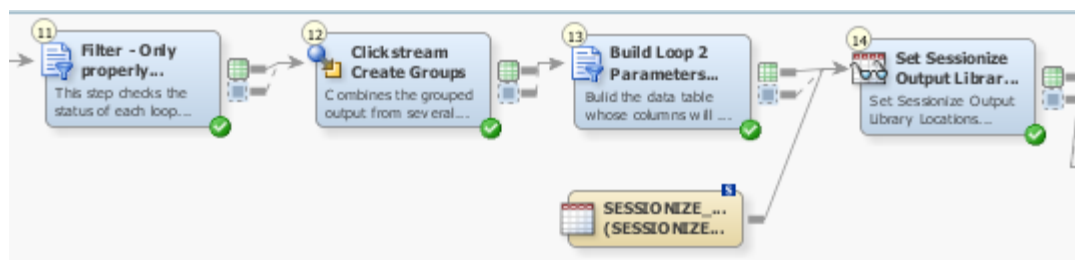
The transformations and tables in this stage are described in the following table:

**Table 9.3** Grouping Transformations

Name	Description	Inputs from and Outputs to
Filter - Only properly parsed logs (SAS Extract) transformation	Uses the status table generated by the Loop transformations to determine which subjobs were successful and should therefore be processed further.	From: Loop 1 (Recognize and Parse) transformation  From: Loop End transformation  To: Clickstream Create Groups transformation
Clickstream Create Groups transformation	Constructs a table that contains information that is used in the sessionize loop; aggregates the parse output groups so that all of the Group 1 session IDs are together, all the Group 2 IDs are together, and so on; prepares views that are ready for the Clickstream Sessionize transformation.	From: Filter - Only properly parsed logs (SAS Extract) transformation  To: Build Loop 2 Parameters (SAS Extract) transformation
Build Loop 2 Parameters (SAS Extract) transformation	Builds a data table that supplies the parameter values for the loop transformation.	From: Clickstream Create Groups transformation  To: Set Sessionize Output Library Locations (Lookup) transformation
SESSIONIZE_GRID_PATHS table	Contains a list of sessionized grid paths.	From: None  To: Set Sessionize Output Library Locations (Lookup) transformation
Set Sessionize Output Library Locations (Lookup) transformation	Assigns each group of tables from the Parse loop to a sessionize output location.	From: Build Loop 2 Parameters (SAS Extract) transformation and SESSIONIZE_GRID_PATHS table  To: Loop 2 (Identify Sessions) transformation

The following display shows the combine groups stage of the template job.

**Display 9.3** Combine Groups Process Flow

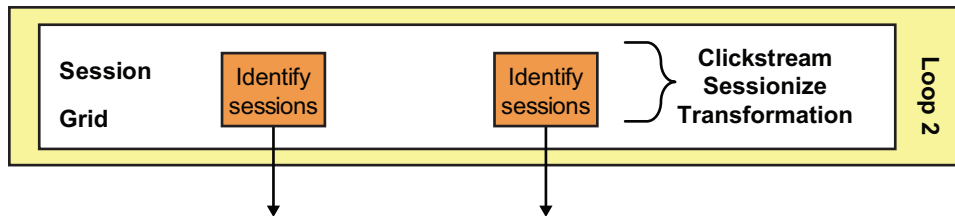


### Loop Two: Sessionize

The fourth stage consists of the second loop. This stage contains transformations and tables that run the loop and sessionize the data.

The following figure illustrates this stage of the process:

**Figure 9.3** Loop Two: Sessionize



The transformations and tables in this stage are described in the following table:

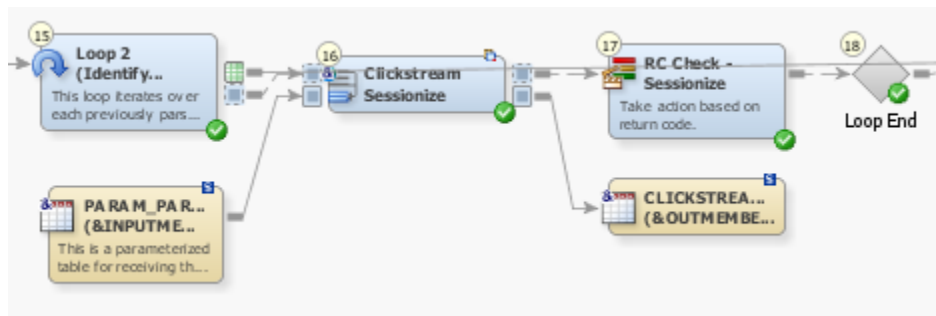
**Table 9.4** Sessionize Transformations

Name	Description	Inputs from and Outputs to
Loop 2 (Identify Sessions) transformation	<p>Sets the parameters that are passed through to the subjobs. The following parameters are set:</p> <ul style="list-style-type: none"> <li>INPUTLIBNAME is the SAS LIBNAME value used to reference all of the output SAS tables from the Clickstream Parse loop.</li> <li>INPUTPATHS is a string formatted for use in the SAS LIBNAME statement. This string specifies the physical paths that contain the SAS table created by the Clickstream Parse loop.</li> <li>INPUTMEMBER is the group of data that is to be processed.</li> <li>OUTMEMBER and OUTLIBPATH define the locations of the Sessionize output.</li> <li>PERMLIBPATH is the path location for the PERMLIB= option; PERMLIB retains data from sessions that were active during processing of the last Web log so that it can continue the sessions later; using PERMLIB enables you to reconnect spanned sessions that were cut when a Web log file ended and a new log file began. The PERMLIB results enable a spanned session to be recognized as the same session by the Clickstream Sessionize transformation.</li> </ul>	<p>From: Set Sessionize Output Library Locations (Lookup) transformation</p> <p>To: Clickstream Sessionize transformation</p> <p>To: Filter Failed Jobs (SAS Extract) transformation</p>
PARAM_PARSE_RESULTS table	<p>A parameterized table for receiving the output from the Clickstream Parse transformation and passing it into the Clickstream Sessionize transformation. (See <a href="#">“Propagating Columns in Jobs that use the Loop Transformation” on page 95</a> if you have defined User Columns that need to be propagated to the final detail table.)</p>	<p>From: None</p> <p>To: Clickstream Sessionize transformation</p>

Name	Description	Inputs from and Outputs to
Clickstream Sessionize transformation	Identifies sessions in the grouped data.	From: Loop 2 (Identify Sessions) transformation and PARAM_PARSE_RESULTS table To: RC Check - Sessionize transformation and CLICKSTREAM_SESSIONIZE table
CLICKSTREAM_SESSIONIZE table	Stores CLICKSTREAM_SESSIONIZE output and ensures the sort sequence of the output data is correct. (See “Backing Up PERMLIB” on page 122.)	From: Clickstream Sessionize transformation To: None
RC Check - Sessionize transformation	Evaluates the return code from the Clickstream Sessionize transformation; sends e-mail to specified address if the sessionized step fails.	From: Clickstream Sessionize transformation To: Loop End transformation
Loop End transformation	Ends loop processing; returns to beginning of loop	From: RC Check - Sessionize transformation To: Filter Failed Jobs (SAS Extract) transformation

The following display shows the second loop stage of the template job.

**Display 9.4** Loop 2 Process Flow

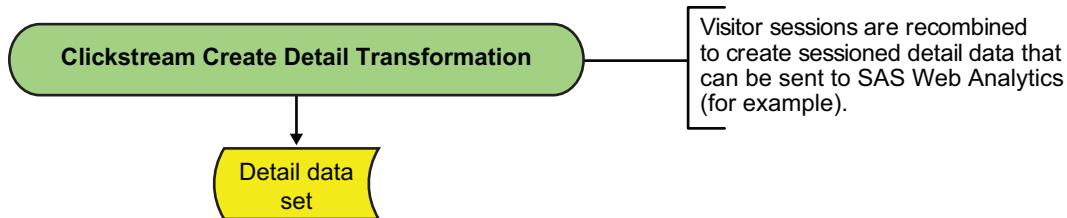


### Create Detail and Generate Output

The fifth stage combines the outputs from multiple Clickstream Sessionize transformations to create a single detail table.

The following display illustrates this stage of the process:

**Figure 9.4** Create Detail and Generate Output



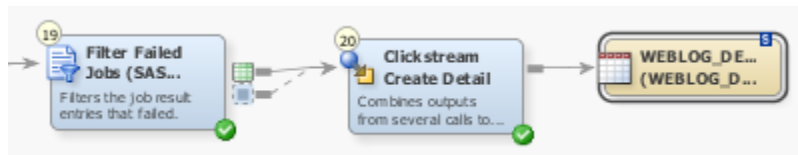
The transformations and tables in this stage are described in the following table:

**Table 9.5** Detail and Output Transformations

Name	Description	Inputs from and Outputs to
Filter - Only properly parsed logs transformation	Uses the status table generated by the Loop transformation to determine which subjobs were successful and should therefore be processed further.	Loop 2 (Identify Sessions) transformation From: Loop End transformation To: Clickstream Create Detail transformation
Clickstream Create Detail transformation	Combines the output from multiple Clickstream Sessionize transformations and creates a single data table.	From: Filter - Only properly parsed logs transformation To: WEBLOG_DETAIL table
WEBLOG_DETAIL table	Contains the output from multiple Clickstream Sessionize transformations.	From: Clickstream Create Detail transformation To: None

The following display shows the create detail and generate output stage of the template job.

**Display 9.5** Create Detail and Generate Output Process Flow



## Running a Basic Template Job

### Overview

Sometimes you want to process the data contained in more than one clickstream log in a single job or improve performance by using parallel processing. In this case, you can

process the job in the appropriate standard Web log template job. Unless you need to process SAS page tags or customer intelligence data, you should use the standard Web logs basic template.

If you have not done so already, you should run a copy of the setup job for the standard Web logs basic template, which is named `clk_0010_weblog_basic_setup`. When you actually process the data, you should run a copy of the standard Web logs job, which is named `clk_0200_weblog_basic_load_weblog_detail`. For information, see [“Copying the Folder Structure of a Clickstream Job” on page 50](#). By running a copy, you protect the original template.

Perform the following tasks to run the template:

- [“Review and Prepare the Job” on page 75](#)
- [“Run the Job and Examine the Output” on page 76](#)

## Review and Prepare the Job

You can examine the standard Web logs basic template job on the **Diagram** tab of the SAS Data Integration Studio **Job Editor** before you run it. You can also configure the job to change the list of logs that you process and set the number of groups that are used in the sessionizing loop. Finally, you can specify parallel and multiple processing options.

Perform the following steps to make these adjustments:

1. Open the renamed standard Web logs basic template job.
2. Scroll through the job on the **Diagram** tab.

Note the following components:

- the two loops and the connections between them
- the transformations that prepare the clickstream logs and groups for loop processing
- the output table that collects the results from the job

For an overview of how the job is processed, see [“Stages in Template Jobs” on page 66](#).

3. Right-click the `Log_Paths` table and select **Open** from the pop-up menu. Review the list of log paths contained in the table. If you need to modify this list, you can click **Switch to edit mode** in the toolbar and make any needed changes.
4. Open the **Loop Options** tabs in the property windows for the two Loop transformations and make sure that the appropriate parallel processing settings are specified. Be particularly careful to ensure that the path specified in the **Location on host for log and output files** field is correct.

For information about the prerequisites for parallel processing, see the “About Parallel Processing” topic in the Working with Iterative Jobs and Parallel Processing chapter in the *SAS Data Integration Studio: User's Guide*. Of course, your job fails if parallel processing has been enabled but the parallel processing prerequisites have not been satisfied.

5. Open the **Parameters** tab in the properties window for the template job and review these two parameters: **Number of Distinct Clickstream Parse Output Paths** and **Number of Groups into which data should be divided**. To access these values, select the parameters and click **Edit** to access the Edit Prompt window. Then, click **Prompt Type and Values** to review the number of groups specified in the **Default value** field. Click **OK** until you return to the **Diagram** tab.

*Note:* The value for these parameters must match the value entered for the setup job. The setup job values are entered on the **Options** tab in the properties window for the Setup transformation in the setup job. If you change either of these values in the template job, you need to rerun the setup job to make sure that the settings match and that the supporting file system structure is generated.

### ***Run the Job and Examine the Output***

Perform the following steps to run a basic job and examine its output:

1. Run the job.



The following display shows the output of a completed job:

**Display 9.6** Completed Basic Job

The screenshot displays the SAS Enterprise Guide interface for a job titled "clk\_0020\_sastag\_basic\_load\_weblog\_detail". The workflow diagram shows the following steps:

- LOG\_PATHS (LOG\_PATHS)**: Contains a list of folder paths to scan for...
- Directory Contents**: Generate a SAS data table containing a...
- Abort if No Files Found**: A decision point that branches based on whether files were found.
- Build Loop Parameters...**: Builds initial loop variables needed in...
- PARSE\_GRID (PARSE\_GRID)**: Contains a list of folder paths across which...

The bottom pane shows the job execution details, including a table of steps and their status.

Order	Name	Status	Details
1	Precode	Completed successfully	
2	Directory Con...	Completed successfully	
3	Abort if No Fil...	Completed successfully	
4	Build Loop Par...	Completed successfully	
5	Set Output Lib...	Completed successfully	
6	Loop 1 (Recog...	Completed successfully	
7	Loop End	Completed successfully	
8	Filter - Only pr...	Completed successfully	
9	Clickstream Cr...	Completed successfully	
10	Build Loop 2 P...	Completed successfully	
11	Set Sessionize...	Completed successfully	
12	Loop 2 (Identi...	Completed successfully	
13	Loop End	Completed successfully	
14	Filter Failed Jo...	Completed successfully	
15	Clickstream Cr...	Completed successfully	
16	Postcode	Completed successfully	
	clk_0020_sast...	Completed successfully	

The status bar at the bottom indicates "Completed successfully".

2. If the job completes without error, right-click the `WEBLOG_DETAIL` table at the end of the job and select **Open** from the pop-up menu.

The View Data window appears, as shown in the following display.

**Display 9.7 Basic Job Output**

#	Entry_Point	Exit_Point	Eyeball_Time	Session_Closed	Status
1	1	0	0:00:04	0	12665
2	0	0	0:00:01	0	12665
3	0	0	0:03:25	0	12665
4	0	0	0:08:20	0	12665
5	0	2		0	12665
6	1	0	0:03:19	0	12675
7	0	0	0:00:01	0	12675
8	0	0	0:03:44	0	12675
9	0	0	0:00:02	0	12675
10	0	2		0	12675
11	1	0	0:05:37	0	12702
12	0	0	0:00:00	0	12702
13	0	0	0:00:02	0	12702
14	0	0	0:00:01	0	12702
15	0	0	0:01:45	0	12702

## Troubleshooting

If the job does not complete successfully, then you might want to examine the logs for each loop in the job. Failures in the main flow of the job are often caused by failures in one or more of the loop SAS sessions. Examine the **Status** tab to determine where the error occurred and refer to the log for that part of the job. A SAS log is saved for each pass through the loops in the Multiple Log Template Job. These logs are placed in a folder called **Process Logs** under the **Loop1** and **Loop2** folders in the structure that is created by the template setup job.

In order to know which file you are looking for, you should understand the naming conventions for these log files. The files in the **ProcessLogs** folder are named `Lnn_x.log`, where `nn` is a unique number for this particular Loop transformation and `x` is a number that represents the iteration of the current loop. For example, if you process 200 Web logs, then the **ProcessLogs** folder for **Loop1** (Clickstream Log transformation and Clickstream Parse transformation) contains 200 logs named `Lnn_1.log` to `Lnn_200.log` (where `nn` is some constant number).

The **ProcessLogs** folder for **Loop2** (Clickstream Sessionize transformation) has the same naming convention. However, the log folder for **Loop2** contains one log for each group. For example, if the Clickstream Parse transformation in the first loop generated five groups, then the logs are named `Lnn_1.log` to `Lnn_5.log` (where `nn` is a constant number).

*Note:* The log files in the **Process Logs** folder are not overwritten during subsequent runs of this job. Consider clearing the **Process Logs** folder between runs during job development to avoid accumulating a large number of logs.

---

## About the Customer Intelligence Template Jobs

You can use the customer intelligence template job to capture information that enables customer Web-based activity to be associated with the marketing campaign that originated the activity. The customer intelligence template job comes in two versions. One version processes standard Web logs and supports campaign data from SAS Digital Marketing. The other handles SAS page tag logs and supports campaign data from SAS Digital Marketing and SAS Real-Time Decision Manager.

The customer intelligence template job enables you to facilitate the collection of the campaign ID so that it is passed to the landing page and remains associated with that customer's session activity. With this approach, analysis can be done that directly attributes actions to campaigns. This analysis can help with determining the success of campaigns and analyzing customers' responses to different types of treatments within a campaign. See [“Processing Campaign Data” on page 91](#) for additional information.

The customer intelligence template jobs are similar to the standard Web log template job. For detailed information about how the Web log template job works, see [“Stages in Template Jobs” on page 66](#).

---

## Collecting Campaign Information in a Customer Intelligence Job

### Overview

The customer intelligence template jobs enable you want to process the data contained in one or more clickstream logs in a single job while filling the marketing campaign information into all records for each job. The campaign information values that you need to capture are the Response Tracking Code (RTC) and the Subject ID (Sn) fields. The following actions support the acquisition of these values:

- New columns have been added to contain campaign information.
- Additional Clickstream Parse rules extract the values from the raw Web log.
- Fill Column options copy the values for these new columns into all records for a session.

This template comes in two versions, one for page-tagged Web logs and another for standard Web logs. The page-tagging version of the template contains the Parse Tagged Data Items transformation in the Loop One: Recognize, Parse, Group Data stage of the job. This transformation is a renamed Clickstream Parse transformation. The topic describes the process that you use to run the standard Web log version of the template job.

You can collect campaign information by using the standard Web logs customer intelligence template job. If you have not done so already, you should run a copy of the setup job for the standard Web logs customer intelligence template job, which is named `clk_0010_weblog_ci_setup`. When you actually process the data, you should run a copy of the customer intelligence template job, which is named `clk_0200_weblog_ci_load_weblog_detail`. By running a copy, you protect the original template. For information, see [“Copying the Folder Structure of a Clickstream Job” on page 50](#).



## **Part 4**

---

# Performing Clickstream Tasks

### *Chapter 10*

**Performing Common Tasks** ..... 83

### *Chapter 11*

**Performing Page Tagging Tasks** ..... 97



## Chapter 10

# Performing Common Tasks

---

<b>Common Clickstream Tasks</b> .....	<b>83</b>
<b>Selecting Files to Process</b> .....	<b>84</b>
<b>Using Character Decoding Options to Process Multi-Byte Data</b> .....	<b>84</b>
Overview .....	84
Using the Decode Option .....	85
Not Using the Decode Option .....	85
Multi-Byte Data Options in Clickstream Transformations .....	85
<b>Filtering Unwanted Data</b> .....	<b>86</b>
<b>Extracting Data from Clickstream Parameters</b> .....	<b>86</b>
Overview .....	86
Create a New Clickstream Parameter .....	87
Other Tasks for Managing Parameters .....	87
<b>Tracking Internal Searches</b> .....	<b>88</b>
Overview .....	88
Populating the int_search_result_page and int_search_term Columns .....	89
Populating the int_search_result_count Column .....	89
<b>Tracking Internal and External Referrers</b> .....	<b>90</b>
<b>Managing Non-Human Visitor Detection</b> .....	<b>90</b>
Overview .....	90
Set Clickstream Sessionize Transformation Options .....	91
<b>Processing Campaign Data</b> .....	<b>91</b>
Overview .....	91
Template Jobs .....	92
Data Model .....	92
Campaign Attribution .....	93
<b>Propagating Columns in Jobs that use the Loop Transformation</b> .....	<b>95</b>

---

## Common Clickstream Tasks

The following tasks are common across all the supplied template jobs:

- [“Selecting Files to Process” on page 84](#)
- [“Using Character Decoding Options to Process Multi-Byte Data” on page 84](#)
- [“Filtering Unwanted Data” on page 86](#)

- “Extracting Data from Clickstream Parameters” on page 86
- “Tracking Internal Searches” on page 88
- “Tracking Internal and External Referrers” on page 90
- “Managing Non-Human Visitor Detection” on page 90
- “Processing Campaign Data” on page 91
- “Integrating SAS Real-Time Decision Manager” on page 100

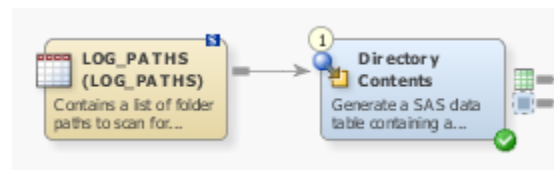
---

## Selecting Files to Process

You need to determine the type of data to process. This selection determines the clickstream template job that you need to use as a basis for processing the data. For information about template selection, see [“Data Source Preparation” on page 15](#).

Once you have determined the files to process, collect them in suitable locations such as a high-performing disk or a network location that optimizes performance. Once you have collected the files, you can enter their locations into the LOG\_PATHS table. This table provides input to the Directory Contents transformation for the template jobs. Note that you can use wildcard filtering and directory recursion on the Directory Contents **Options** tab. After you change the settings, run the Directory Contents transformation only and view the output table to ensure that you have selected the desired set of files. The following display shows the LOG\_PATHS table and Directory Contents transformation:

**Display 10.1** File Selection Objects




---

## Using Character Decoding Options to Process Multi-Byte Data

### Overview

Two types of decoding occur during the ETL process for SAS Data Surveyor for Clickstream Data. It is important that you fully understand the implications in order to ensure the best data quality. The following decoding types are available:

#### URL Encoding

URL encoding is also known as percent encoding. This encoding type is a mechanism used to encode certain reserved characters (or unsafe characters) that might appear in a URL. These reserved characters might have special meaning if not otherwise encoded. For example, a form on a Web page could have a text field where customers can leave a short comment that includes a question mark character. Submitting this form results in the following URL:



```
/submit.php?text=Is this really true?
```

A question mark is a reserved character in a URL that is used to separate the requested file from the query string. In order to remove any confusion as to which question mark is the separator, the URL is automatically encoded as follows:

```
/submit.php?text=Is%20this%20really%20true%3F
```

This URL represents how the URL would be encoded. The space character is encoded as `%20` and the non-reserved question mark is encoded as `%3F`.

### Character Encoding

Character encoding is required when the URL sent to the Web server contains characters that are not part of the all US-ASCII character set. For example, a Japanese customer who performed a search on the word Rugby would type in the characters ゾグビー. These characters, if passed as part of the URL, would be encoded as `%E3%83%A9%E3%82%B0%E3%83%93%E3%83%BC`. The actual percent encoding values would depend on the character set used in the Web page.

As a user, you should consider the following topics:

- “Using the Decode Option” on page 85
- “Not Using the Decode Option” on page 85

## Using the Decode Option

By default, the template jobs have decoding turned on. Furthermore, the incoming encoding of the Web log is set to UTF-8. The primary benefit of decoding the incoming data is that it becomes readable by humans. Because the data has been decoded, you no longer see encoded values in filtered data or reports. For information about the SAS Unicode Server, see “Using a Unicode SAS Application Server” on page 41.

## Not Using the Decode Option

You should not turn on decoding when you process a standard Web log that contains records from pages that used different encodings or character sets. Character set information for each record is not present in a standard Web log. Therefore, all of the records are assumed to use the same encoding. Any Web pages with encoding that does not match the one selected for the incoming data to this ETL will not be decoded correctly. SAS page tag logs do not have this problem because the page code automatically encodes all data records using UTF-8.

## Multi-Byte Data Options in Clickstream Transformations

The following table covers decoding options in Clickstream transformations:

**Table 10.1** Multi-Byte Data Options in Clickstream Transformations

Option	Location	Setting	Notes
Decode input data?	Input pane of the <b>Options</b> tab in the Clickstream Log transformation	<b>Yes</b>	Determines whether decoding should be performed for each line of data as it is read from the input. When set to the default value of <b>Yes</b> , you must set the encoding of the incoming data correctly and use a SAS Unicode Application Server (if necessary).

Option	Location	Setting	Notes
Encoding of the incoming data	Input pane of the <b>Options</b> tab in the Clickstream Log transformation	Not displayed unless Decode input data? is <b>Yes</b>	Specifies the original encoding of the data in the input file. All SAS page tag logs are encoded in UTF-8 irrespective of the encoding used on the originating Web page. However, you must ensure that all incoming records come from Web pages with the same encoding when you process standard Web logs. Otherwise, data could be transcoded incorrectly.
Decode referrer ?	Input pane of the <b>Options</b> tab in the Clickstream Log transformation	<b>No</b>	Determines whether the data contained in the referrer is encoded as it is processed. The referrer field has to be treated as a separate decoding entity because the encoding of the referrer string is often unknown since it might come from an external site. If the encoding of all the referrers within your data is known, then this option can be set to <b>Yes</b> . Otherwise, it should remain set to <b>No</b> .
Sort options	Table pane of the <b>Options</b> tab in the Clickstream Sessionize transformation	Blank	Specifies sort options that can be useful for changing the linguistic collation of output when MBCS data is processed.  See PROC SORT documentation for details about valid syntax and available options.

---

## Filtering Unwanted Data

You can filter data on the Clickstream Parse transformation's **Rules** tab. A rule is designed to examine incoming records to determine whether a certain condition is met. Then the rule performs a specified action in response to the met condition. To summarize, you can filter unwanted data through the following process:

1. Define a rule with the condition set to the desired filter criteria.
2. Set the action to **Delete record**.

---

## Extracting Data from Clickstream Parameters

### Overview

You want to store the value from an incoming cookie, query, or referrer parameter in a user column.

You can use the **Clickstream Parameters** tab in the properties window for the Clickstream Parse transformation. Each row in the table on the **Clickstream Parameters** tab identifies

a parameter that is parsed from the log during processing. Furthermore, each parameter can be assigned to a user column that stores the parameter's value.

The row for each parameter consists of the following columns:

**Name**

specifies the name of the column in the table.

**Description**

describes the contents of the parameter.

**Source Type**

identifies the source type from which the parameter is parsed. The available source types are None, Cookie, Query, and Referrer.

**User Column**

specifies the name of the variable that stores the parsed parameter's data value.

Perform the following tasks to manage parameters:

- [“Create a New Clickstream Parameter” on page 87](#)
- [“Other Tasks for Managing Parameters” on page 87](#)

## Create a New Clickstream Parameter

Perform the following steps to create a new clickstream parameter:

1. Use the **User Columns** tab to ensure that an appropriate user column has been created that will accept the value of the clickstream parameter that you intend to parse.
2. Click **Create a new parameter** on the **Clickstream Parameters** tab to add a row to the parameters table.
3. Enter a name and description for the new parameter in the Name and Description columns. The name entered must exactly match the actual name of the parameter as it exists in the Web log for which the value is being captured. For example, query and referrer parameters are often found in the Web log as name value pairs (Q=Value), with pairs often delimited by & characters (Q=Value1&R=Value2&S=Value3). In this case, the Name of the three parameters would be Q, R, and S, respectively.
4. Select a source type from the drop-down menu in the Source Type column.
5. Select a user column from the drop-down menu in the User Column column.

## Other Tasks for Managing Parameters

For information about other ways to manage parameters, such as copying, importing, or exporting a parameter, see the Help for the **Clickstream Parameters** tab.

*Note:* If you create many clickstream parameters, it is a good practice to export them for reuse. By default, clickstream parameters are exported to an XML file in your Windows user folder. You cannot selectively export clickstream parameters; all are exported. However, you can select individual parameters when you import them.

Consider the following factors when you export and import clickstream parameters:

- The easiest way to determine which parameters exist in a Web log is to save the UNIQUEPARMS table to a permanent location and import clickstream parameters from this physical table. Open the properties window for the Clickstream Parse transformation and select the **Tables** pane on the **Options** tab. You can then specify an **Additional output library**. The UNIQUEPARMS table will be saved to this

location after processing a Web log. You have the option to rename it by changing the value of the **Unique parameters output table** option on this same tab.

- After you import parameters from the UNIQUEPARMS table, you must create or import corresponding user columns. After you create or import appropriate user columns, you must select a user column for each parameter on the **Clickstream Parameters** tab. A UNIQUEPARMS data table has no previously defined user column assignments for parameters. Instead, it is simply a list of all parameters available in that Web log. This fact explains why the value of **Description** is set to **Untitled** after initial import.
- Consider creating or importing any user columns that you need before you import any previously exported clickstream parameters from an XML file. Otherwise, the user columns assignments are missing and you must manually reselect the corresponding user columns for all imported clickstream parameters.

## Tracking Internal Searches

### Overview

Internal searches are performed on a page within a Web site. These searches look for content within that Web site. That is, the search is made for results that are found internally within the site.

In order to effectively track internal searches and provide for reporting on those searches, the columns listed in the following table have been added to the output detail data set:

**Table 10.2** Internal Search Output Columns

Column Name	Default Value	Description
int_search_result_page	0	Contains a value of 1 if the corresponding Requested_File being accessed is a page that contains search results.
int_search_term	Missing	Search string for which the user submitted a query to the search page.
int_search_result_count	Missing	Count of the number of search results found. Only available when you use the page-tagging template job.

Note that the SAS page tag tutorial and template jobs output all three of the internal search output columns. Note that standard Web logs do not contain the information about search results needed to set int\_search\_result\_count. Therefore, the Standard Web Log tutorial and template jobs only output int\_search\_result\_page and int\_search\_term.

You can enable these internal search output columns by performing the following tasks:

- [“Populating the int\\_search\\_result\\_page and int\\_search\\_term Columns” on page 89](#)
- [“Populating the int\\_search\\_result\\_count Column” on page 89](#)

### Populating the `int_search_result_page` and `int_search_term` Columns

The `int_search_result_page` column specifies the page that contains the search results and the `int_search_term` specifies the query string parameter used for the search term. You can set up these columns in the **Rules** tab of the Clickstream Parse transformation for the template jobs by editing the rules in the Search group.

The settings that you need to make are listed in the following table:

**Table 10.3** Settings for the `int_search_result_page` and `int_search_term` Rules

Option	Location	Setting	Notes
Enable Set <code>int_search_result_page</code> rule	Clickstream Parse ( <b>Rules</b> tab)	Yes (in Enable column)	None
Set Column search	Clickstream Parse ( <b>Rules</b> tab)	Search.php (default)	For the Set <code>int_search_result_page</code> rule, access the Rule Properties window. Then, click <b>Search Options</b> to review or change the default.
Set Assign column	Clickstream Parse ( <b>Rules</b> tab)	Set Column to <code>int_search_result_page</code> and Value to 1	Also in the Rule Properties window.
Add query string parameter containing the internal search term	Clickstream Parse ( <b>Clickstream Parameters</b> tab)	Set Name to <code>q</code> and User Column to <code>int_search_term</code>	Add a new Clickstream Parameter, setting <b>Name</b> to the parameter that contains the search term ( <code>q</code> in this example), and <b>User Column</b> to <code>int_search_term</code> .

### Populating the `int_search_result_count` Column

The `int_search_result_count` (ISC) column displays the number of search results found when you enable it in a page-tagging template job. This column is available only in tutorial and template jobs that are processing a SAS page tag data source. This function is automatically handled by the SAS page tag template job, provided that the tagging data element ISC is captured within the tag code. The SAS page tag setup job automatically generates test data containing this element. Perform the following steps to add ISC to the data elements being captured by the tag:

1. Ensure that your search results page is tagged.
2. Add JavaScript code to the `st_pageDats()` method on the search results page that retrieves the number of search results. Then it sets the value of the ISC data element to this number. For example, you can add the following code:

```
function st_pageDats()
{
    // Add code here which retrieves the number of search results
    result_count="100";

    // Set ISC value to the number of search results on page load (0x1)
    st_rq.dats.add( "ISC", result_count, true, 0x1);
}
```

This step passes the value of ISC passed from the SAS page tag to the collection server. When the SAS page tag Web log is processed from the collection server by the SAS page tag template job, this job automatically parses the value of ISC into the `int_search_result_count` column of the job's output table.

---

## Tracking Internal and External Referrers

It is sometimes useful to know whether the referring domain is from a page that is internal to the site (an internal referrer) or from a page outside of the site (an external referrer). You can use the `referrer_internal` column to track whether a referrer is internal or external.

When this column is set to a value of *I*, the referrer is understood to be an internal referrer. Otherwise, the referrer is understood to be an external referrer.

In the template jobs, the **Rules** tab in the Clickstream Parse transformation contains a sample rule called **Detect internal referrer** for setting the `referrer_internal` column value to *I*. This rule is disabled by default. When enabled, it searches the `Referrer_Domain` to determine whether the record was referred by an internal domain.

---

## Managing Non-Human Visitor Detection

### Overview

Spiders, robots, crawlers, pingers, and any other computer program that might generate traffic to a Web site are referred to as non-human visitors (NHV). Spiders (a search engine bot, for example) surf the Web site traveling various links to determine the contents of all of the Web pages. All spiders or NHVs have certain behavior characteristics that make it possible to identify them. These characteristics include clicking at a rate faster than humanly possible or pinging at an exact interval.

Activity from NHVs is handled in two locations. The first is in the Clickstream Parse transformation using the Filter Spiders by User Agent rule. This rule matches commonly known strings found in the user agent of well-behaved NHVs who identify themselves as an NHV. By default, this rule deletes activity for these NHVs. The purpose of this detection is to eliminate NHV clicks as soon as possible.

The second location of NHV activity is handled during the Clickstream Sessionize transformation. The transformation uses a proprietary behavioral detection approach that examines the behavior of the visitor within a session and decides whether the behavior is likely to be that of a human or a non-human visitor. This process is known as Behavioral Identification of Non-Human Sessions (BINS), and is configured using the spider-related options on the Clickstream Sessionize transformation. See the Clickstream Sessionize **Options** tab help for details about how to configure this functionality.

If you have already filtered and removed the NHVs found by the Clickstream Parse transformation using the rule that examines the User Agent string, you might want to analyze the visitor behavior to ensure that none of the remaining sessions were created by NHVs. To perform this analysis, set the options in the Clickstream Sessionize properties window to detect any NHVs.

### Set Clickstream Sessionize Transformation Options

Perform the following steps to set the options in the Clickstream Sessionize properties window:

1. Open the **Tuning** category on the **Options** tab in the Clickstream Sessionize Properties window.
2. Specify a value in the **Spider detection threshold**, **Spider force threshold**, and **Maximum average time between spider clicks** fields. For example, the Web site's administrator determines that for the site's visitors, no human visitor is likely to perform more than 50 clicks in a session. Therefore, you might decide to set the **Spider force threshold** to 50, forcing the detection of an NHV when the number of clicks in the session reaches 50 or higher.
3. Select a value in the **Spider Action** field. This value determines whether the session is isolated, deleted, or no action is taken once the spider is identified.

Although the Spider Action does not directly impact the detection of NHVs, it does impact what happens to the data for any NHV. The default of **ISOLATE** is useful as it separates the non-human data into a separate table and enables you to validate that the detection heuristics are accurate. This table is found in the library specified by the **Additional output library** option in the Tables group in the Clickstream Sessionize transformation **Options** tab. The **DELETE** action is perhaps useful once the heuristics are considered accurate and you just want the non-human data discarded. The final option of **NONE** means that the non-human sessions are not identified, so they are treated as any other session data.

---

## Processing Campaign Data

### Overview

You can now associate Web activity with the marketing campaign that originated that activity.

This enhanced capability comes with new functions built into SAS Data Surveyor for Clickstream Data and new and revised template jobs that include the following:

- Built-in support for tracking of campaigns originating from SAS Digital Marketing (SDM)
- Built-in support for tracking of campaigns originating from SAS Real-Time Decision Manager (RTDM)
- Integration with SAS Real-Time Decision Manager. This integration enables for dynamic campaign content (a treatment) to be loaded to a Web page based on information specific to that session made in real time. Any subsequent activity as a result of the treatment maintains an association with it. This association can help you

determine the success of campaigns and analyze customers' responses to different types of treatments within a campaign.

These functions have been dynamically integrated with the SAS Real-Time Decision Manager. Therefore, you must ensure that the Web pages have been instrumented with the SAS page tag and that a clickstream collection server is active.

The ability to process campaign data depends on the following components:

- “Template Jobs” on page 92
- “Data Model” on page 92
- “Campaign Attribution” on page 93

## Template Jobs

Two Customer Integration (CI) template jobs are provided out of the box. These jobs are located in the **Products** ⇒ **SAS Data Surveyor for Clickstream Data** ⇒ **2.2** ⇒ **Template Jobs** folder in the SAS Data Integration Studio **Folders** tab. The Customer Intelligence template job contained in the **SAS Page Tag Logs** folder supports both SAS Digital Marketing and SAS Real-Time Decision Manager campaigns. The Customer Intelligence job in the **Standard Web Logs** folder supports only campaigns from SAS Digital Marketing.

## Data Model

Columns are available in the data model to integrate with SAS Customer Intelligence platform. You can use the Rules and Parameter Parsing functionality of the Clickstream Parse transformation to automatically populate several columns with values.

The available columns are covered in the following table:

**Table 10.4** Columns in the Data Model

Column Name	Label	Description	Notes
EntrySource	Entry Source	An ID that represents the entity that originated the access to the landing page	Automatically populated with <b>SDM</b> for SAS Digital Marketing or <b>RTDM</b> for SAS Real-Time Decision Manager using rules defined in the Clickstream Parse transformation.  SAS Web Analytics can also populate this column with values to represent other sources such as AdWords.
EntryActionID	Entry Action ID	An ID that represents the item from the Entry Source that originated the access to the landing page	When the EntrySource is set to <b>SDM</b> , this column is set to the value that occurs in the RTC= query parameter value. When EntrySource is set to <b>RTDM</b> , this column is set to the value that occurs in the RTDMRTC= SAS page tag parameter value.
EntryTreatmentID	Entry Treatment ID	An ID that represents a particular action (a treatment) within a campaign that was clicked on or opened by the subject	When the EntrySource is set to <b>RTDM</b> , this column is set to the value that occurs in the RTDMTTC= SAS page tag parameter.



Column Name	Label	Description	Notes
S1	Subject ID Parm1	Identifies the subject of an Entry Action either alone or with other Subject ID parameters	Value obtained from the appropriate query parameter
S2	Subject ID Parm2	Identifies the subject of an Entry Action either alone or with other Subject ID parameters	Value obtained from the appropriate query parameter
S3	Subject ID Parm3	Identifies the subject of an Entry Action either alone or with other Subject ID parameters	Value obtained from the appropriate query parameter
S4	Subject ID Parm4	Identifies the subject of an Entry Action either alone or with other Subject ID parameters.	Value obtained from the appropriate query parameter
EntryPlacement	Campaign placement	Describes the position of the campaign creative element that result in the click-through of a Web site	Values have to be defined by the user.
EntryCreative	Campaign creative	The creative element that resulted in the click-through of a Web site	Values have to be defined by the user.

## Campaign Attribution

The **Input** pane in the **Options** tab of the Clickstream Sessionize transformation contains the **Forward Fill Columns** and **Complete Fill Columns** options. These options can be used to propagate values that occur in one record within a session through to other records for which the column does not contain a value. The **Forward Fill Columns** option only propagates values into subsequent session records. The **Complete Fill Columns** option propagates values into preceding and subsequent session records.

The example data below demonstrates the **Forward Fill Columns** function. A customer visits the main page of an online store and then remembers receiving a promotional e-mail. The customer clicks on a link in the promotional e-mail. As a result, the Web log records a record with the RTC and Sn parameters. The customer then proceeds through the site and ultimately makes a purchase, as shown in the following:

```

2001-07-01 0:00:00 202.192.163.153 - GET /main/index.html ...
2001-07-01 0:01:05 202.192.163.153 - GET /store/music/d.asp?rtc=123456789
&s1=a1 &s2=a2 &s3=a3 &s4=a4 ...
2001-07-01 0:01:20 202.192.163.153 - GET /store/music/artist.asp ...
2001-07-01 0:02:35 202.192.163.153 - GET /store/music/albums.asp?artist=241
2001-07-01 0:03:14 202.192.163.153 - GET /store/music/detail.asp?album=241-001 ...

```

Typically, the session data would be constructed as described in the following table:

**Table 10.5** Campaign Session Data

Datetime	Client IP	URI	QueryString	Entry Source	Entry ActionID	S1	S2	S3	S4
2001-07-01:00:00	202.192.163.153	/main/index.html							
2001-07-01:01:05	202.192.163.153	/store/music/d.asp	rtc=123456789 &s1=a1 &s2=a2 &s3=a3 &s4=a4	SDM	123456789	a1	a2	a3	a4
2001-07-01:01:20	202.192.163.153	/store/music/artist.asp							
2001-07-01:02:35	202.192.163.153	/store/music/albums.asp	?artist=241						
2001-07-01:03:14	202.192.163.153	/store/music/detail.asp	?album=241-001						

Adding the EntrySource, EntryActionID, and S1 to S4 columns to the Forward Fill Columns dialog box results in the values being propagated through the subsequent records of this session. This enables us to attribute the final result of a user's session with a campaign. The result is shown in the following table:

Datetime	Client IP	URI	QueryString	Entry Source	Entry ActionID	S1	S2	S3	S4
2001-07-01:00:00	202.192.163.153	/main/index.html							
2001-07-01:01:05	202.192.163.153	/store/music/d.asp	rtc=123456789 &s1=a1 &s2=a2 &s3=a3 &s4=a4	SDM	123456789	a1	a2	a3	a4
2001-07-01:01:20	202.192.163.153	/store/music/artist.asp		SDM	123456789	a1	a2	a3	a4
2001-07-01:02:35	202.192.163.153	/store/music/albums.asp	?artist=241	SDM	123456789	a1	a2	a3	a4
2001-07-01:03:14	202.192.163.153	/store/music/detail.asp	?album=241-001	SDM	123456789	a1	a2	a3	a4

---

## Propagating Columns in Jobs that use the Loop Transformation

SAS Data Integration Studio can automatically propagate columns from one transformation to another. However, this propagation is not necessarily possible in template jobs that use the Loop and Loop End transformations.

Therefore, any user columns that are created in the Clickstream Log or Clickstream Parse transformation in the first loop do not automatically appear in the final detail output table. Two approaches to manual propagation are available. Use the first approach if the user column has not been defined yet, and use the second if it has.

The first approach recommends that users add a user column that has not been previously defined to the final destination table, which is typically a permanent table such as a final detail table. Then, you can import the column into the other locations where it is required. If you add the column to the final destination table, you must import the column into the appropriate locations in the job.

For example, you might need to perform the following tasks:

- Open the **Columns** tab in the properties window for a PARAM\_PARSE\_RESULTS table in a job. Then import the desired column from the final detail table. The column is automatically propagated to the Clickstream Sessionize target.
- Import the column into the **User Columns** tab in Clickstream Log or Clickstream Parse transformations in the first loop in the job. Then, click the **Target Table** tab in Clickstream Parse to add the column to the target table.

Perform the following steps to manually propagate a user column that has already been defined in the first loop and is output from the Clickstream Parse transformation:

1. Add the column to the PARAM\_PARSE\_RESULTS table in the second loop, which is an input to the Clickstream Sessionize transformation. Once added, it is automatically propagated to the target table of this transformation.
2. Add the column to the final detail table.



## Chapter 11

# Performing Page Tagging Tasks

---

<b>Configuring Link Tracking in Tagged Pages</b> .....	<b>97</b>
Overview .....	97
Tracking Links By File Extension .....	98
Tracking Links By ID .....	98
Tracking Links By Name .....	99
Tracking Links By Other Attributes .....	99
<b>Integrating SAS Real-Time Decision Manager</b> .....	<b>100</b>
Overview .....	100
Details .....	100

---

## Configuring Link Tracking in Tagged Pages

### Overview

You can modify the JavaScript tagging code for your site or your tagged pages to support the tracking for specific types of links. These code modifications only take effect if you accept the default setting of `st_cfg.cap['L']=""`; which enables the tracking of all links. This default setting is also sufficient for the link tracking needed for the following scenarios:

#### Tracking Off-Site Links

Off-site links are present. These links are found when you have tagged pages of interest for your organization's site but your site also includes links that direct the user to another organization's Web site. Data about when these links are accessed is normally missing because it is not possible to tag the other organization's Web pages. You want to know when these links are accessed, but you do not want to use tagged redirect pages as a detection mechanism because of the maintenance overhead.

#### Tracking Non-Tagged Intra-Site Links

Non-tagged intra-site links are present. These links are found when you have tagged pages of interest for your organization's site. However, your site also includes links that direct the user to another department or subsite within the organization that cannot be tagged. Data about when these links are accessed is normally missing because it is not possible to tag the other department or subsite's Web pages. You want to know when these links are accessed, but you do not want to use tagged redirect pages as a detection mechanism because of the maintenance overhead.

### Tracking Links to Non-Taggable Content

Links to non-taggable content are present. These links are found when you have tagged pages of interest for your organization's site. However, your site also includes links from pages on the organization's site that access content that cannot itself be tagged (such as PDF and XLS). You want to know when these links are accessed, but you do not want to use tagged redirect pages as a detection mechanism because of the maintenance overhead. In addition to the default tracking of all links, you can use the approach described in [“Tracking Links By File Extension” on page 98](#).

*Note:* The following features are available only in the maintenance release of SAS Data Surveyor for Clickstream Data 2.1:

- the ability to track clicks on links based on attributes other than the file extension of link target
- the ability to track clicks on links that leave the Web site

The following scenarios require you to use enter code under the `st_trk` function:

- [“Tracking Links By File Extension” on page 98](#)
- [“Tracking Links By ID” on page 98](#)
- [“Tracking Links By Name” on page 99](#)
- [“Tracking Links By Other Attributes” on page 99](#)

### Tracking Links By File Extension

Use the `st_cfg.cap['L']` array element to specify the file types that you need to track. For example, you can track only the links to PDF, DOC, and XLS files with the following code: `st_cfg.cap['L']="0:pdf:doc:xls";`

*Note:* Setting `st_cfg.cap['L']` to a non-blank value turns off the use of `st_trk`. If there are other conditions besides tracking by file extension to consider when determining whether a link should be tracked, do not use this approach. Instead, set `st_cfg.cap['L']=""` and write code for each condition to be checked in `st_trk`.

### Tracking Links By ID

You can limit your tracking to specific links on a given page and base the tracking decision on the ID of the link. This approach enables you to avoid gathering tracking data for all of the other links on the page. For example, you can track the IDs of two links, such as `linkID1` and `linkID2`.

To implement this approach, ensure that `st_cfg.cap['L']="";` has been entered to enable link tracking. Then, define the `st_trk` function in the `SASSiteConfig.js` file as follows:

```
function st_trk
{
  switch(o.nodeName.toLowerCase())
  {
    case 'a':          // Link elements
      if (o.id=='linkID1') return true;
      if (o.id=='linkID2') return true;
      return false;
      break;
    default:
      return true;
  }
}
```

```

    }
}

```

## Tracking Links By Name

You can limit your tracking to specific links on a given page and base the tracking decision on the name of the link. This approach enables you to avoid gathering tracking data for all of the other links on the page. For example, you can track the names of two links, such as `linkName1` and `linkName2`.

To implement this approach, ensure that `st_cfg.cap['L']="";` has been entered to enable link tracking. Then, define the `st_trk` function in the `SASSiteConfig.js` file as follows:

```

function st_trk
{
    switch(o.nodeName.toLowerCase())
    {
        case 'a':           // Link elements
            if (o.name=='linkName1') return true;
            if (o.name=='linkName2') return true;
            return false;
            break;
        default:
            return true;
    }
}

```

## Tracking Links By Other Attributes

You can limit your tracking to specific links on a given page. Then you can base the tracking decision on an attribute that you have defined and placed into the HTML for the links to track. This approach enables you to avoid gathering tracking data for all of the other links on the page.

For example, you can track links with a `TRACKME` attribute set to 1. With this attribute implemented, a link in the format `<A HREF="http://www.sas.com">SAS</A>` would change to `<A HREF="http://www.sas.com" TRACKME=1>SAS</A>`

To implement this approach, ensure that `st_cfg.cap['L']="";` has been entered to enable link tracking. Then, define the `st_trk` function in the `SASSiteConfig.js` file as follows:

```

function st_trk
{
    switch(o.nodeName.toLowerCase())
    {
        case 'a':           // Link elements
            if (o.attributes['TRACKME'].value=='1') return true;
            return false;
            break;
        default:
            return true;
    }
}

```

---

## Integrating SAS Real-Time Decision Manager

### Overview

You can combine the functionality provided by SAS Data Surveyor for Clickstream Data and SAS Real-Time Decision Manager. This combination enables you to dynamically update Web page content targeted to the customer in real time. Then, it lets you record and track any response as a result of activity on the modified page. This approach provides a way to dynamically present targeted content to a customer. It also facilitates establishing the effectiveness of the campaign.

The SAS page tag functionality passes session information through an asynchronous request to the SAS Real-Time Decision Manager Web service, which responds with a targeted response. The response contains information about a treatment that should be displayed on the current Web page. For example, the treatment can identify an image contained in the content management system that should be displayed on the Web page. The SAS page tag functionality then updates the Web page source to display the appropriate content. If the customer then clicks on this treatment (which is typically a link), information about the campaign that generated the treatment is recorded in the SAS page tag log and later processed as part of the ETL.

See [“Processing Campaign Data” on page 91](#) for information about the type of data collected and how it is processed. See [“Configuring SAS Real-Time Decision Manager Integration” on page 31](#) for information about configuring your Web pages to interact with SAS Real-Time Decision Manager Web service.

### Details

The following sequence of events and different components are involved in the integration of SAS Data Surveyor for Clickstream Data and SAS Real-Time Decision Manager:

1. When a Web page loads that is instrumented with the SAS page tag RTDM module, it passes session information (this could be a customer ID) to the designated RTDM Web service.
2. The Web service returns a response that contain information about what should be displayed on the page, such as a graphic and the link to be associated with the graphic.
3. The SAS page tag RTDM module uses the information to update the targeted component on the Web page.
4. If the customer then clicks the link generated by the RTDM Web service, the customer is taken to the target page and information of the click. The various tracking codes for the RTDM treatment are recorded in the clickstream collection server log.
5. The SAS Data Surveyor for Clickstream Data ETL processes the resulting SAS page tag log and generates a detail data set.
6. SAS Web Analytics can be used to incorporate information from the common data model to generate campaign-related reports.



## Part 5

---

# Clickstream Transformations

<i>Chapter 12</i>	
<b>Log Transformation</b> .....	103
<i>Chapter 13</i>	
<b>Parse Transformation</b> .....	109
<i>Chapter 14</i>	
<b>Sessionize Transformation</b> .....	119
<i>Chapter 15</i>	
<b>Specialized Transformations</b> .....	127



## Chapter 12

# Log Transformation

---

<b>About the Clickstream Log Transformation</b> . . . . .	<b>103</b>
<b>Specifying the Path to the Log</b> . . . . .	<b>104</b>
Overview . . . . .	104
Specify the Path to the Web Log . . . . .	105
<b>Maintaining Log Types</b> . . . . .	<b>105</b>
Overview . . . . .	105
Maintain Log Types . . . . .	106
<b>Managing User Columns</b> . . . . .	<b>106</b>
Overview . . . . .	106
Create a New User Column . . . . .	107
Reuse User-Defined Columns in Other Clickstream Jobs . . . . .	107
Other Tasks for Managing User Columns . . . . .	107
<b>Specifying Log Options</b> . . . . .	<b>107</b>

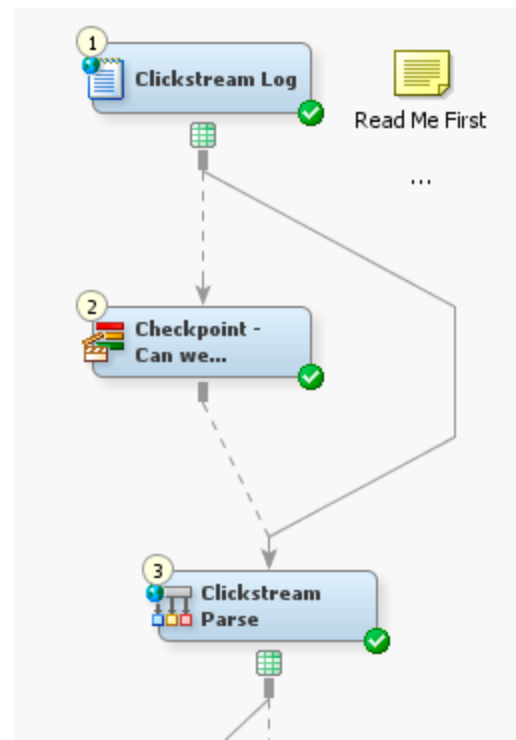
---

## About the Clickstream Log Transformation

The Clickstream Log transformation reads a clickstream log and checks the format of the log against the log formats or log types that are enabled on the **Log Types** tab in the properties window for the transformation. If there is a match, the transformation maps the columns from the log to the Clickstream Parse Input Columns and loads an output table with data from the log. This table becomes the input to a Clickstream Parse transformation.

The following display shows the Clickstream Log transformation in one of the tutorial jobs that are provided with the SAS Data Surveyor for Clickstream Data.

**Display 12.1** Clickstream Log Transformation in the Standard Web Log Tutorial Job



Typical user tasks for the Clickstream Log transformation include the following:

- specifying the physical path to the Web log
- working with log type definitions
- adding or modifying user-defined columns
- setting options for the Clickstream Log transformation

---

## Specifying the Path to the Log

### Overview

In the SAS Data Integration Studio Job Editor, you have opened a job that includes the Clickstream Log transformation. You want to specify the path to the Web log to be processed. You can use the **File Location** tab in the properties window for the Clickstream Log transformation to specify the location of the Web log. (This approach assumes that you are working with a copy of one of the template jobs, or that you have dragged and dropped the Clickstream Log transformation into another job.)

## Specify the Path to the Web Log

Perform the following steps to specify the path to the Web log:

1. Right-click the Clickstream Log transformation. Then select **Properties** ⇒ **File Location** tab.
2. Select or type the path to the Web log file. The path must be accessible to the SAS Application Server that executes the job. If you specify a path, you can use the **Preview** button to have the SAS Application Server attempt to retrieve the first few lines of the file. This is helpful to validate that you have specified the path correctly.

You can specify the filename as a path or a SAS macro variable, such as **&INPUTFILE**. A SAS macro variable might be useful if you use the Clickstream Log transformation in a loop. For loop processing, the current value of the SAS macro variable is used. Note that all template jobs use loop processing. However, tutorial jobs do not use loop processing.

---

## Maintaining Log Types

### Overview

You can use the **Log Types** tab in the properties window for the Clickstream Log transformation to view, add, or update the log type definitions. Each row in the table on the **Log Types** tab represents the metadata for a log type.

#### Enable

specifies whether a log type is enabled for the current transformation. **Yes** means that the log type is enabled. To enable or disable a log type, double-click the value in this field and use the selection arrow to select a different value.

*Note:* If you do not plan to process logs in a particular format, then disable the corresponding log type. The Clickstream Log transformation no longer checks for that log type. Disabling unused log types can reduce the time that the transformation spends on detecting the format of a log.

#### Name

specifies a unique identifier for the log type such as SASTAG or IPLANET. These identifiers are collected in the SAS code that is generated when the Clickstream Log transformation runs.

#### Description

describes the log type. The default log types are as follows:

- SAS Tag Data Format (SAS page tag log from the Clickstream collection server)
- Sun iPlanet Log Format (iPlanet Netscape)
- NCSA Common Combined Log Format (CLFE)
- W3C Extended Log Format (ELF)

*Note:* The order in which the log types appear on the **Log Types** tab is important. It reflects the order that is used to identify the type of the incoming Web log. If an incoming log matches more than one log type, ensure that the more specific comparison is performed first (higher on the list of log types). For example, a

Web log that matches the SAS Tag Data format log type also matches the NCSA Common Combined Log Format (CLFE). Accordingly, the SAS Tag Data format is listed first.

## Maintain Log Types

To work with the detailed metadata for a log type, select its row on the **Log Types** tab, and then click an appropriate toolbar option. Alternatively, you can right-click the row for a log type and select an appropriate option from the pop-up menu. Unique options for log types include the following:

### Create a new log type

adds a row with default settings for a log type. To update the detailed metadata for the new type, select the new row and use the toolbar or the pop-up menu to select **Properties**.

### Create a copy of the selected log type

copies the metadata for the selected log type and adds the copy to the end of the list. You can then update the copy to create a new log type that is similar to the one you copied. Note that you might want to reorder the log types to recognize your new log type first. You can also disable the log types that you no longer want to process in this job. Refer to the Help for the options under the **Log Types** tab for more details.

### Properties

displays the detailed metadata for the selected log type. Use this option to view or update attributes for a log type, such as the mapping between input columns from the log and output columns for the current transformation.

### Import log types

enables you to select and import an XML file that specifies a set of log types that were exported from the **Log Types** tab.

*Note:* Currently, when you import log types, you import all log types that were exported. This might result in duplicate copies of the default log types. Duplicates should be deleted.

### Export log types

enables you to export all log types on the **Log Types** tab to an XML file.

*Note:* Currently, an export operation exports all log types that are displayed on the **Log Types** tab, including the default log types.

---

## Managing User Columns

### Overview

Sometimes an input column from the Web log does not have a matching Clickstream Parse Input Column. In that case, you can use the **User Columns** tab in the properties window for the Clickstream Log transformation to add a user-defined column. The new column appears in the output table for the Clickstream Log transformation and is available to Clickstream transformations later in the process flow for the job.

Each row in the table on the **User Columns** tab contains the metadata for a user column.

The row for each user column consists of the following columns:

**Name**

specifies the name of the column in the table.

**Description**

specifies a description for the contents of the column.

**Type**

specifies the data type of the column.

**Length**

specifies the length of the column.

**Format**

specifies the SAS format used (if needed) to specify formats for the selected column.

**Is Nullable**

if present, indicates whether a column can contain null or missing values.

Perform the following tasks:

- [“Create a New User Column” on page 107](#)
- [“Reuse User-Defined Columns in Other Clickstream Jobs” on page 107](#)
- [“Other Tasks for Managing User Columns” on page 107](#)

### **Create a New User Column**

Perform the following steps to create a new user column:

1. Click the **New column** button in the toolbar to add a row to the table on the **User Columns** tab.
2. Enter appropriate values in the Name, Description, Type, Length, Format, and Is Nullable columns.

### **Reuse User-Defined Columns in Other Clickstream Jobs**

The **User Columns** tab lists the metadata for any user-defined output columns that have been defined for the Clickstream Log transformation or the Clickstream Parse transformation. You cannot export user-defined columns from the **User Columns** tab and then import them into other jobs. However, you can use the background pop-up menu on any output table in the job that contains the user columns and then register the output table. Any user-defined columns in these tables can then be imported into the **User Columns** tab, using the **Import Columns** option on that tab.

### **Other Tasks for Managing User Columns**

For information about other ways to manage user columns, see the Help for the **User Columns** tab.

---

## **Specifying Log Options**

Use the **Options** tab in the properties window for the Clickstream Log transformation to set options that are not set in the other tabs in the window. For example, you can use the

**Maximum detection lines** field in the **Input** pane to specify the maximum number of lines to read when attempting to determine the log type. If the log type is not detected after reading the number of input records specified by this option, then the clickstream log is not processed.

For information about the other options on the tab, see the Help for the **Options** tab.



## Chapter 13

# Parse Transformation

---

<b>About the Clickstream Parse Transformation</b> .....	<b>109</b>
<b>Best Practices for the Clickstream Parse Transformation</b> .....	<b>111</b>
Handling Non-Human Visitors in the Clickstream Parse Transformation .....	111
Maintaining the Hold Buffer Size Setting .....	111
Optimizing Sort Using SORTSIZE .....	111
Setting a Visitor ID Value .....	112
<b>Identifying Incoming Columns</b> .....	<b>112</b>
Overview .....	112
Maintain Column Mappings .....	113
Other Tasks for Input Mapping .....	113
<b>Maintaining User Columns</b> .....	<b>113</b>
Overview .....	113
Create a New User Column .....	114
Reuse User-Defined Columns in Other Clickstream Jobs .....	114
Other Tasks for Managing User Columns .....	114
<b>Applying Clickstream Parse Rules</b> .....	<b>114</b>
Overview .....	114
Create a New Rule .....	115
Other Tasks for Managing Rules .....	116
<b>Managing the Visitor ID</b> .....	<b>116</b>
Overview .....	116
Select a User Column as the Visitor ID .....	116
<b>Managing Output Table Columns</b> .....	<b>117</b>
Overview .....	117
Specify the Output Columns from the Clickstream Parse Transformation .....	117
<b>Specifying Parse Options</b> .....	<b>117</b>

---

## About the Clickstream Parse Transformation

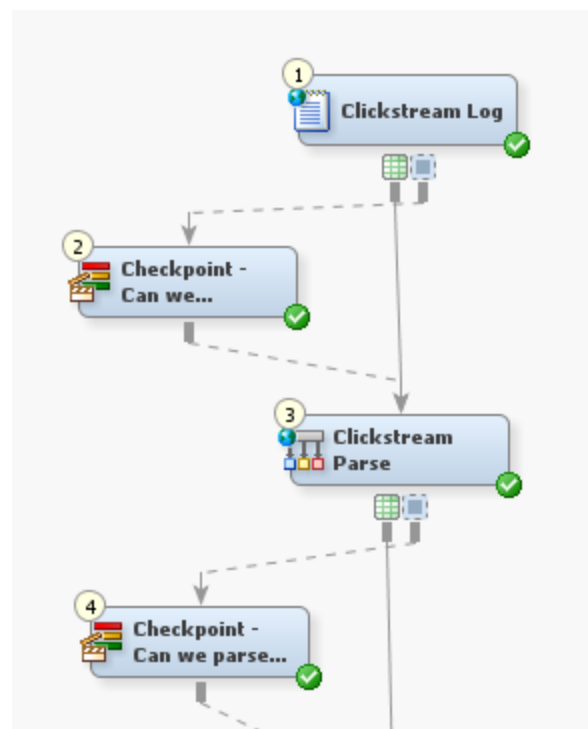
The Clickstream Parse transformation usually reads the output table from the Clickstream Log transformation. However, the Clickstream Parse transformation can read from any input table containing clickstream data that can be mapped to standard clickstream parse input columns. Then, it interprets this incoming data and creates a common set of output columns, independent of the incoming Web log type.

The Clickstream Parse transformation performs the following functions:

- associates input columns with “Clickstream Parse Input Columns” on page 137 that have specific roles during processing
- filters unwanted data records from the target table, according to both default and user-defined rules.
- enables the definition of one or more cookie, query string, or referrer parameters to be parsed and stored as new data items in the target table.
- if possible, uniquely identifies the visitor who is associated with each data record and adds the visitor ID as a new data item in the target table.
- creates an output table used as the input to the Clickstream Sessionize transformation.
- provides several built-in rules for processing incoming data, as well as several sample rules.
- stores the value from an incoming cookie, query, or referrer parameter in a user column. For information, see “Extracting Data from Clickstream Parameters” on page 86.

The Clickstream Parse transformation is shown in the following display.

**Display 13.1** Clickstream Parse Transformation



---

## Best Practices for the Clickstream Parse Transformation

### *Handling Non-Human Visitors in the Clickstream Parse Transformation*

Spiders, robots, crawlers, pingers, and any other computer programs are referred to as *non-human visitors* (NHVs). Activity from an NHV is handled by two approaches. The first approach uses the **Filter Spiders by User Agent** rule in the Clickstream Parse transformation. This rule matches commonly known strings found in the user agent of well-behaved NHVs that identify themselves as NHVs. By default, this rule deletes activity for these NHVs. The purpose of this detection is to eliminate NHV clicks as soon as possible.

In the second approach, NHV activity is handled in the Clickstream Sessionize transformation. The Clickstream Sessionize transformation uses a proprietary behavioral detection approach called Behavioral Identification of Non-Human Sessions (BINS). This approach examines the behavior of the visitor within a session. Then, it decides whether the behavior is more likely to be that of a human or a non-human visitor. For more information, see [“Managing Non-Human Visitor Detection” on page 90](#).

### *Maintaining the Hold Buffer Size Setting*

The option entered in the **Hold Buffer Size** field in the **Input** pane on the **Options** tab in the Clickstream Parse transformation can have a significant effect on the performance of the transformation. When Web servers write raw data to the logs, the records are typically written in chronological order. The hold buffer size option represents the amount of this data that is held in memory before it is written to the output table.

For example, the default value of **120** causes all records that have a timestamp within the last 120 seconds of the latest timestamp to be held in memory. With this value, any records that have a date-and-time stamp that is not within that 120-second range are added to the output table. This hold buffer usually enables any incoming records that are slightly out of chronological order to be corrected. Thus, a subsequent sort of the data can generally be avoided.

However, the default hold buffer size does not always work as expected. If you find that your incoming data is out of chronological order and exceeds this 120-second threshold, you can increase the hold buffer size. However, the larger hold buffer increases the memory used by the Clickstream Parse transformation because more data is held in the buffer before it is sent to the output table.

If the hold buffer functionality is consistently unable to prevent a sort, it can be switched off with a value of **0**. This setting can result in a subsequent sort being required. However, it removes some of the processing overhead that occurs in managing the buffer.

### *Optimizing Sort Using SORTSIZE*

The Clickstream Parse transformation attempts to reorder records with the hold buffer size option to avoid a sort. However, a sort might be required in cases where records are severely out of chronological order. By default, the Clickstream Sessionize transformation performs a final sort of the output with Session ID, Date and Time, and Record ID set as the sort criteria. Minimizing the need to perform disk I/O operations improves the performance.

In both cases, performance can be improved by setting the SORTSIZE option in SAS. This option can be set on the **Precode and Postcode** tab found in all transformations. Simply select the **Precode** check box and enter a SORTSIZE value in the field such as **options SORTSIZE=2G;**. This code sets the SORTSIZE to 2 GB of RAM.

If you are running in an SMP or grid environment (such as in a multiple log job), keep in mind that this setting applies for each parallel process. For example, if you are running on a four-processor machine with 16 GB of total RAM, setting SORTSIZE to 2.5 GB consumes up to 10 GB of RAM (2.5 x 4 processors). This setting leaves 6 GB for the operating system and any other processes running on the machine.

### Setting a Visitor ID Value

Whenever possible, select the columns that contain the visitor ID value on the **Visitor ID** tab of the Clickstream Parse transformation. A good visitor ID value uniquely identifies the activity of one and only one visitor. Thus, the quality and accuracy of the sessionized data is enhanced significantly when a known visitor ID value is provided.

Therefore, you should avoid using the default algorithm based on the client IP address and user-agent string, although this might be the only option available in some scenarios. For information about selecting a visitor ID, see the Help for the **Visitor ID** tab. Also, see [“Managing the Visitor ID” on page 116](#).

---

## Identifying Incoming Columns

### Overview

When you want to identify the meaning of the incoming columns to the Clickstream Parse transformation, maintain the column mappings between the source tables from the Clickstream Log transformation (or any other previous transformation) and the Clickstream Standard Input Columns Table. If the column name in the source table matches a Clickstream Log Standard Column, then the mapping is performed automatically. For information about the Clickstream Standard Input Columns (Clickstream Parse Input Columns), see [“Clickstream Parse Input Columns” on page 137](#).

You can maintain column mappings on the **Input Mapping** tab in the properties window for the Clickstream Parse transformation. The source column data comes from the Clickstream Log transformation (or the output table of any previous transformation or another input table) that precedes the Clickstream Parse transformation in the process flow.

User columns defined in the Clickstream Log transformation should be mapped on the **Input Mappings** tab in the Clickstream Parse transformation when they are intended to serve in the role of a Clickstream Parse Standard Input Column. Otherwise, their values can be used in a rule or simply passed through to the Clickstream Parse transformation target table using the **Target Table** tab.

*Note:* You can also define user columns on the **User Columns** tab on the Clickstream Parse transformation and tie them to parameters on the **Clickstream Parameters** tab (also on the Clickstream Parse transformation). Also, you can create user columns based on the rules that you create on the **Rules** column on the Clickstream Parse transformation. The user columns created on these tabs are added to the output on the **Target Table** tab on the Clickstream Parse transformation. These user columns do not appear on the **Input Mapping** tab.

The **Column assignments for** list box on the **Input Mapping** tab contains Clickstream Parse Standard Input Columns. If you add a column on the **User Columns** tab of the Clickstream Log transformation or the Clickstream Parse transformation, you can map it here.

Perform the following tasks:

- [“Maintain Column Mappings” on page 113](#)
- [“Other Tasks for Input Mapping” on page 113](#)

## Maintain Column Mappings

The **Input Mapping** tab contains a set of tools to map between the output columns of the prior transformation and the input columns of the Clickstream Parse transformation.

Perform one of the following tasks to map between the input and output columns:

- Click **Map all columns** to map between all of the input and output columns. Columns are automatically matched when the column name of a source table matches a column name in the output table.
- Click **Map selected columns** to map between a set of columns that you have selected and the appropriate output columns.

## Other Tasks for Input Mapping

For information about other ways to manage input mapping, such as building an expression for a derived mapping or deleting a mapping, see the Help for the **Input Mapping** tab.

---

# Maintaining User Columns

## Overview

You can use the **User Columns** tab in the properties window for the Clickstream Parse transformation when you want to define your own columns. These user columns can be used to store interim values during the parse process or they can be added to the target table on the **Target Table** tab.

Specifically, the users columns often serve the following purposes:

- holding values determined by user-defined rules on the **Rules** tab
- storing values from clickstream parameters defined on the **Clickstream Parameters** tab.

Each row in the table on the **User Columns** tab contains the metadata for a user column.

The row for each user column consists of the following columns:

### Name

specifies the name of the column in the table.

### Description

specifies a description for the contents of the column.

### Type

specifies the data type of the column.

**Length**

specifies the length of the column.

**Format**

specifies the SAS FORMAT used (if needed) to specify formats for the selected column.

**Is Nullable**

if present, indicates whether a column can contain null or missing values.

Perform the following tasks

- [“Create a New User Column” on page 114](#)
- [“Reuse User-Defined Columns in Other Clickstream Jobs” on page 114](#)
- [“Other Tasks for Managing User Columns” on page 114](#)

**Create a New User Column**

Perform the following tasks to create a new user column:

1. Click **New column** to add a row to the user columns table.
2. Enter appropriate values in the Name, Description, Type, Length, Format, and Is Nullable columns.

**Reuse User-Defined Columns in Other Clickstream Jobs**

The **User Columns** tab lists the metadata for any user-defined output columns that have been defined for the Clickstream Log transformation or the Clickstream Parse transformation. You cannot export user-defined columns from the **User Columns** tab and then import them into other jobs. However, you can use the background pop-up menu on any output table in the job that contains the user columns and then register the output table. Any user-defined columns in these tables can then be imported onto the **User Columns** tab, using the **Import Columns** option on that tab.

**Other Tasks for Managing User Columns**

For information about other ways to manage user columns, such as copying, importing, or modifying a parameter, see the Help for the **User Columns** tab.

---

## Applying Clickstream Parse Rules

**Overview**

You can use the **Rules** tab in the properties window for the Clickstream Parse transformation to perform specified record-level processes based on the clickstream input data stored in a record.

Common record-level processes include the following tasks:

- filtering data
- conditionally assigning a value to a variable
- executing custom SAS code

Each row in the table on the **Rules** tab consists of a criteria and an action that is associated with the criteria that is matched within the data. The tools on the tab enable you to develop a set of rules associated with an instance of a Clickstream Parse transformation in a process flow. Apply these rules to each record that is processed by the transformation.

For example, you can use a rule to search for and remove unwanted data records from a source table. This approach saves the significant records in a target table for continued analyses. If you are analyzing a marketing campaign, you can detect and remove records that contain unwanted ZIP codes from your target table. Then, they would not be included in future runs of the analysis.

The row for each rule consists of the following columns:

**Enabled**

specifies whether the rule is active (Yes or No) for the current transformation.

**Group**

specifies the name of the group to which the rules belong. Typical default names for groups include Samples, Filters, and User.

**Name**

specifies the name of the rule. Typical default names for rules include Filter local IP address, Filter graphic files, Filter spiders by user agent, User code after input, and User code after parse.

**When**

specifies the stage in the parse process at which the rule is applied. The default values are After input and After parse.

**Condition Type**

specifies criteria against which a data record is tested. The valid condition testing methods are Always, Column Search, SAS expression, and Regular expression.

**Action Type**

specifies the activity to perform when the condition is met. The valid actions are Delete, Assign, and Code.

Perform the following tasks to manage rules:

- [“Create a New Rule” on page 115](#)
- [“Other Tasks for Managing Rules” on page 116](#)

## Create a New Rule

Perform the following tasks to create a new rule:

1. Click **Create a new rule** to add a row to the rules table.
2. Enter the name of the group for the rule in the Group column.
3. Enter a name for the rule in the Name column.
4. Enter appropriate values in the When, Condition Type, and Action Type columns.
5. Click **Properties** to access the Rule Properties window. Enter the appropriate conditions and actions for the new rule.
6. Click **OK** to close the Rule Properties window.

## Other Tasks for Managing Rules

For information about other ways to manage rules, such as copying, importing, or exporting a rule, see the Help for the **Rules** tab.

If you create many customized rules, you should export them to an XML file for import into other jobs that require the same rules to process other Web logs. By default, this XML file goes to your local Windows user folder location and not to the machine where your workspace server is running (unless your workspace server is located on your local machine). You cannot selectively import and export rules at this time. All rules are processed, or none are processed. Therefore, you also export all of the sample rules provided with the product when you export the rules that you create. After you import the XML file that contains the rules into another job, you receive a second set of default rules that can be deleted at that time.

---

## Managing the Visitor ID

### Overview

You can use the **Visitor ID** tab to identify one or more user columns to be used for setting the value of the visitor ID. Perform this task when you want to manage the identification of the visitors to the Web sites listed in clickstream logs. A clickstream log is a collection of entries (one entry per click) that is made by multiple visitors to a Web site. Evaluation of clickstream log files can provide business analysts with useful information about each visitor's behavior during Web site visits.

Each person who uses a Web browser to access a Web site is considered a visitor of that Web site. Web developers use clickstream parameters (such as a cookie or a query parameter) to uniquely identify visitors to their Web site. These clickstream parameters can be parsed into user columns, which can then be assigned to the visitor ID. A visitor ID is designed to contain a unique value for each visitor.

The **Available** list box contains the list of valid user columns for visitor ID values. You can use the **User Columns** tab to define a user column. Then, you can use the **Clickstream Parameter** tab to map a clickstream parameter to a user column.

Selecting the **Set visitor ID to the last non-blank column below** check box causes the Clickstream Parse transformation to parse the list of user columns in order and use the first non-blank value to populate the visitor ID value. If all user columns are blank, then the default algorithm using the client IP and user agent string is used to populate the visitor ID.

### Select a User Column as the Visitor ID

Perform the following tasks to select a user column as the visitor ID:

1. Select the **Set visitor ID to the last non-blank column below** check box to activate the list box functions on the tab.
2. Select an appropriate user column in the **Available** list box.
3. Click the arrow between the list boxes to move the column to the **Selected** list box. Note that you can also select a column in the **Selected** list box and return it to the **Available** list box.



---

## Managing Output Table Columns

### Overview

Sometimes you want to select the columns that are present in the output from the Clickstream Parse transformation. In this case, you can use the **Target Table** tab in the properties window for the Clickstream Parse transformation.

You can select additional output columns from the following sources:

- The Clickstream Log Table lists all input columns in the source table for the Clickstream Parse transformation. To simply pass through an input column to the output, select the column from this list in the **Available columns** list box. Then move it to the **Selected columns** list box.
- The Clickstream Parse Standard Columns lists all possible Clickstream Standard Output Columns. This source includes new output columns that are generated by the Clickstream Parse transformation.
- The Clickstream Parse User Columns Table lists all user columns that are defined on the **User Columns** tab.

*Note:* You must add any user columns that you want to include in the target table to the **Selected columns** list box on the **Target Table** tab. The user columns that you define are not automatically included in the target table.

### Specify the Output Columns from the Clickstream Parse Transformation

Perform the following steps to specify the output columns from the Clickstream Parse transformation:

1. Select the columns that you want to add to the output table from the columns listed in the **Available columns** list box.
2. Click the arrow between the list boxes to move the columns to the **Selected columns** list box. Note that you can also select columns in the **Selected columns** list box and return them to the **Available columns** list box.

---

## Specifying Parse Options

Use the **Options** tab in the properties window for the Clickstream Parse transformation to set options that are not set in the other tabs in the window. For example, you can specify the number of groups in a multiple log job in the **Number of groups** field in the **Grouping** pane on the tab. You can also specify query, cookie, and URI delimiters in the **Delimiters** pane and adjust the hold buffer size in the **Input** pane. (For more information about the hold buffer size, see [“Maintaining the Hold Buffer Size Setting” on page 111.](#))

For information about the other options on the tab, see the Help for the **Options** tab.



## Chapter 14

# Sessionize Transformation

---

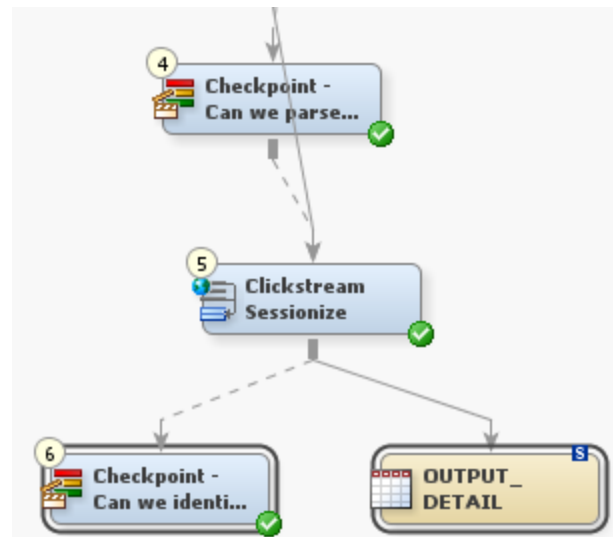
<b>About the Clickstream Sessionize Transformation</b> . . . . .	<b>119</b>
<b>Best Practices for the Clickstream Sessionize Transformation</b> . . . . .	<b>122</b>
Backing Up PERMLIB . . . . .	122
Managing the Contents of PERMLIB . . . . .	122
<b>Visitor ID Completion</b> . . . . .	<b>122</b>
Overview . . . . .	122
Process . . . . .	123
<b>Spanning Web Logs</b> . . . . .	<b>123</b>
<b>Specifying Options for the Sessionize Transformation</b> . . . . .	<b>123</b>
Input Options . . . . .	123
Tables Options . . . . .	124
Tuning Options . . . . .	124

---

## About the Clickstream Sessionize Transformation

The Clickstream Sessionize transformation reads data from the input transformation (typically the Clickstream Parse transformation). Once the input data is clean and you have identified a Visitor ID, then you need to identify sessions. A session consists of the series of the user's clicks from the time that the user enters the Web site, clicks on certain pages, and then exits at another point.

The Clickstream Sessionize transformation enables you to identify the user sessions, identify spiders and other non-human visitors, and manage sessions that span Web logs. The output goes to a table or continues within the job for additional processing. The Clickstream Sessionize transformation is shown in the following display.

**Display 14.1** Clickstream Sessionize Transformation

The Clickstream Sessionize transformation passes the same set of columns that it receives on the input to the output. The transformation also adds the following columns:

**Table 14.1** Clickstream Sessionize Generated Columns

Column Name	Description	Completion Method	Label	Length	SAS Format
Session_ID*	Specifies the assigned session identifier for this visitor session.  *Default name for the column identified as holding or representing the Session ID.	If the Session_ID column is present on the input table and has a value, then this value is used as the identifier for this visitor's session. If the Session_ID value is blank or the Session_ID column is not present in the incoming table, then it is derived from User-Defined Rules or the default configuration option. (This option combines CLK_Client_IP, CLK_cs_UserAgent, and date time.)	Session ID	245	\$245
Session_Closed	Specifies whether the record belongs to an open or closed session. When this value is set to <b>1</b> , it indicates that this record belongs to a closed session. A value of <b>0</b> indicates that this record belongs to an open session.	Is set to <b>1</b> when a session has exceeded the session timeout value. Otherwise, this value is set to <b>0</b> .	Session Closed	3	1.

Column Name	Description	Completion Method	Label	Length	SAS Format
Entry_Point	Specifies whether this is the first click of the visitor's session. When this value is set to <b>1</b> , it indicates that this is the first click of the visitor's session. Otherwise, this value is set to <b>0</b> .	Examines in date_time order. The first click entry_point is set to <b>1</b> ; all others are set to <b>0</b> .	Entry Point	3	1.
Exit_Point	Specifies whether this is the last click of the visitor's session and whether it belongs to an open or closed session. When this value is set to <b>1</b> , it indicates that this is the last click of the visitor's session and it belongs to a closed session. When this value is set to <b>2</b> , it indicates that this is the last click of the visitor's session and it belongs to an open session. Otherwise, this value is set to <b>0</b> .	Examines clicks in date_time order. The final click exit_point is set to <b>1</b> ; all others are set to <b>0</b> or <b>2</b> .	Exit Point	3	1.
Eyeball_Time	Specifies the amount of time the visitor spent on the page before the next click.	Subtracts date_time of current click from date_time of subsequent click. The last click in a session is set to missing.	Eyeball Time	8	TIME.

*Note:* Extra columns that are on the input to the Clickstream Sessionize transformation are passed through. The generated columns are added to the output detail data table.

Typical user tasks for the Clickstream Sessionize transformation include the following:

- specifying the way that non-human visitors are detected and handled
- managing sessions that span Web logs
- specifying options for the Clickstream Sessionize transformation

---

## Best Practices for the Clickstream Sessionize Transformation

### ***Backing Up PERMLIB***

When the Clickstream Sessionize transformation has completed execution, any sessions that are still considered open have data stored in a permanent library that has the default libref, PERMLIB. This information is used to process the next Web log in a series when user sessions span across Web logs. The next time the Clickstream Sessionize transformation executes, if possible, a user's open session data in PERMLIB will be combined with session data for that user in the current run. In this way, a complete record for a user can be captured across different runs of the clickstream job.

Make sure you back up the contents of PERMLIB before each execution of the Clickstream Sessionize transformation in a clickstream job. If the Clickstream Sessionize transformation should fail for some reason, and the tables in PERMLIB are in an unknown state, then the backup can be restored and the job can be rerun. The physical path to PERMLIB is specified in the library definition on the **Options** tab in the properties window for the Clickstream Sessionize transformation, in the **Tables** section.

### ***Managing the Contents of PERMLIB***

If you use a clickstream job to reprocess the same data multiple times, as you might do when developing a new clickstream job, be sure to return PERMLIB to the state that it was in before the last execution. If PERMLIB was empty, then its contents should be removed. If PERMLIB contained tables, then the tables should be restored to the state that they were in before the last execution. Otherwise, duplicate data appears in the output table for the Clickstream Sessionize transformation.

Use the **Delete PERMLIB Tables** option in the **Tables** section of the **Options** tab in the Clickstream Sessionize transformation to control how PERMLIB is managed. When you set the job up, this option is set to delete PERMLIB tables. When you are ready to run the job in production, this option should be changed so that session data is correctly connected across executions of the job. See the Help for the Delete PERMLIB Tables option from the options tab.

---

## Visitor ID Completion

### ***Overview***

One important function the Clickstream Sessionize transformation performs is Visitor ID completion. Visitor ID completion copies the visitor ID (once known) to the other data records in the session for which the visitor ID is missing. Because every click of the session now contains a valid visitor ID, you can analyze this visitor activity to determine the visitor's original referring site. For example, this can be useful in determining how much revenue (during cart check-out) was generated from the referring site. This information can help determine whether advertising dollars are being well spent.

The value for the Visitor ID is configured using the **Visitor ID** tab of the Clickstream Parse transformation. The Clickstream Sessionize transformation then performs Visitor ID completion on any records for which Visitor ID was not assigned a value.

## Process

No user steps are required, but this is how the process works:

- A customer (visitor) visits Web site A, but has not logged in.
- The first data record of their session contains the name of the site (site B) that referred the customer to site A. However, it does not contain the ID of the visitor.
- After several clicks, none of which contain the visitor ID, the visitor logs in.
- Most of the subsequent clicks now contain a valid Visitor ID.
- The visitor checks out having made a purchase.
- The visitor logs out.
- The visitor clicks several more pages, but no clicks contain the visitor ID.

Visitor ID completion copies the known visitor ID to the other data records in the session where the visitor ID was missing. A complete session is created for better analysis.

---

## Spanning Web Logs

If a session extends from one Web log in to another, the data collected on that session from the first Web log is incomplete. In this case, the Clickstream Sessionize transformation cannot determine whether the session is complete, and the record is marked with a **2** in the **Exit point column** field of the output record. This incomplete session data is held in the permanent library, which was set in the **Permanent library path** field on the **Options** tab of the Clickstream Sessionize transformation. Once the following day's data is captured in another Web log, the session data is matched up and collected. For example, if the cutoff for the Web log is at midnight, but a user clicks on that Web site from 11:30 p.m. until 12:30 a.m., then the session information is contained in two Web logs. The data from the first day is held in the permanent library as incomplete until it is matched with the second Web log. This is why it is important to properly manage the content of the Permanent library path between runs. See [“Best Practices for the Clickstream Sessionize Transformation” on page 122](#).

---

## Specifying Options for the Sessionize Transformation

### Input Options

Use the **Options** tab to identify the input columns (typically Clickstream Parse), whose values are used by the Clickstream Sessionize transformation. In the **Input** window, you set the options to identify which of the input columns should be used in the various roles required for the Clickstream Sessionize transformation to operate properly. For example, **Visitor ID column** uniquely identifies the visitor. If no Visitor ID is available, an algorithm based on values from the **Client IP column** and the **User agent column** is used to create

a new Visitor ID. This combined value is a last resort, as the value of analytics performed without a reliable visitor ID is severely reduced. Column roles for record ID, date, and timestamp are set here as well.

Additional input options are available to support propagation of values both forward and completely through the visitor session. For additional information about input options, see the Help for the Clickstream Sessionize **Options** tab.

## Tables Options

The **Tables** options window is used to set the characteristics of the output table, specify new or existing columns, and specify libraries to be used.

The most commonly used table options include the following:

- The **Sort output table**, **Sort sequence**, and **Sort** options enable management of how the output data is ordered.
- **Additional output library** stores output data, including records of non-human interactions such as those done by spiders.
- **Permanent library path** is used when the session is not closed because that user's session carried over into the next day's run, which was captured in a separate Web log. These sessions are marked with a **2** in the **Exit point column** field. (See the **Exit point column** description in the following list.)
- **Delete PERMLIB tables** deletes PERMLIB tables when the job is initially set up. PERMLIB is used to store records from visitor sessions that have not yet closed. These represent activity of visitors occurring at the time that the Web log file was closed, and a new file started. Normally, the option should be set to **Yes** when you develop and test a job, running the same data multiple times. You should set the option to **No** when your job functionality has stabilized because batches of Web log data must be processed in chronological order.
- **Session ID column** creates a new column that identifies a particular user's session.
- **Entry point column** is a binary field that represents whether this click is where the user entered the Web site.
- **Exit point column** is a field that represents whether this click is where the user exited the Web site. The values can be **0** (Not an exit point), **1** (Is an exit point), and **2** (Do not know yet — pending the next 30 minutes of data to determine).
- **Eyeball time column** is the amount of time the user spent on a page before continuing to the next page.
- **Session Closed column** indicates whether the session is completed.

For additional information about table options, see the Help for the Clickstream Sessionize **Options** tab.

## Tuning Options

The **Tuning** options window is used to determine session, group, and spider characteristics and how to handle them. The most commonly used tuning options include the following:

- **Session timeout** determines the amount of time of inactivity until the session is closed. By default this is set to 30 minutes, which is an industry standard. However, you can change this value if you determine that there is a more appropriate value. If there is no activity for a particular visitor for 30 minutes or more, then the visitor's session is



determined to have ended. If the time-out value has expired and activity restarts, a new session starts and is given a new Session ID.

- **Spider detection threshold**, **Spider force threshold**, and **Maximum average time between spider clicks** are used to identify non-human activities.
- **Spider detection threshold** controls the minimum number of clicks that must be in a session before NHV detection is performed on that session.
- **Spider force threshold** controls the number of clicks in a session after which classification of the session as an NHV session is forced.
- **Maximum average time between spider clicks** controls the maximum average spacing between click activity in a session under which the session is classified as an NHV session.
- **Spider Action** is used to determine how to handle spider sessions once they are identified.

As with any tuning option, you should experiment with the settings to achieve the desired results for your data. The combination of these options determines the number of spiders detected. The basic reactions are in the following list:

- Raising the **Maximum average time between clicks** detects more spiders.
- Lowering the **Spider detection threshold** detects more spiders.
- Raising the **Spider detection threshold** detects fewer spiders.
- Lowering the **Spider force threshold** detects more spiders.
- Decreasing the **Session timeout** value results in more sessions.

For additional information about any of these or other Sessionize options, see the Help for the Sessionize **Options** tab.



## Chapter 15

# Specialized Transformations

---

<b>About the Specialized Transformations</b> .....	<b>127</b>
<b>Directory Contents Transformation</b> .....	<b>128</b>
<b>Clickstream Setup Transformation</b> .....	<b>128</b>
<b>Clickstream Create Detail Transformation</b> .....	<b>128</b>
<b>Clickstream Create Groups Transformation</b> .....	<b>128</b>

---

## About the Specialized Transformations

The SAS Data Surveyor for Clickstream Data provides several transformations that perform specialized functions for the job in which they are used.

This section provides information about the following transformations:

**Table 15.1** *Specialized Transformations*

Transformation	Description
Directory Contents transformation	Generates a SAS data table that contains a listing of the files found in a path or list of paths, and if selected, their subfolders. It is used in the template jobs as described in <a href="#">“Prepare Data and Parameter Values to Pass to Loop 1” on page 66</a> .
Clickstream Setup transformation	Generates the folder structure on the file system to hold the SAS logs and any generated data files. It also generates configuration data if necessary and test Web log data for the tutorial and template jobs. Used in Clickstream Setup (_setup) jobs.
Clickstream Create Groups transformation	Combines the grouped output from several calls to the Clickstream Parse transformation into a set of output views, one per group. It is used in the template jobs as described in <a href="#">“Combine Groups” on page 70</a> .
Clickstream Create Detail transformation	Combines the output from multiple Clickstream Sessionize transformations and creates a single data table. It is used in the template jobs as described in <a href="#">Figure 9.4 on page 74</a> .

---

## Directory Contents Transformation

You can use the Directory Contents transformation to create a SAS table containing the list of files found in either a single directory or a list of directories. The Directory Contents transformation can be found in the Access folder in the SAS Data Integration Studio Transformations tree.

When the Directory Contents transformation contains an input table, it reads the list of directories to scan from the entries in that table. Otherwise, the directory to scan is obtained from the single directory specified by the **Input directory to scan** option on the **Options** tab of the Directory Contents transformation. See the Help for the **Options** tab for more information.

---

## Clickstream Setup Transformation

You can use the Clickstream Setup transformation to create the directory structure and test data needed for a specific tutorial or template job. This transformation is typically used within a `_setup` job associated with a tutorial or template job. This `_setup` job should be run before you run the main processing job in a tutorial or template job.

You can find the Clickstream Setup transformation in the Clickstream Transformations folder in the SAS Data Integration Studio Transformations tree. When you run the Clickstream Setup, it creates a folder structure and test data based on the options specified on the **Options** tab of the transformation. See the Help for the **Options** tab for more information.

---

## Clickstream Create Detail Transformation

You can use the Clickstream Create Detail transformation to combine the output from a loop that has run several iterations of the Clickstream Sessionize transformation.

You can find the Clickstream Create Detail transformation in the Clickstream Transformations folder in the SAS Data Integration Studio Transformations tree. When you run the Clickstream Create Detail transformation, it creates a single output detail data table based on the options specified on the **Options** tab. This output data table is typically the final output from a clickstream job, and it is consumed by later analytic processes or solutions (such as SAS Web Analytics). See the Help for the **Options** tab for more information.

---

## Clickstream Create Groups Transformation

You can use the Clickstream Create Groups transformation to combine the grouped output from a loop that has run several iterations of the Clickstream Parse transformation.

You can find the Clickstream Create Groups transformation in the Clickstream Transformations folder in the SAS Data Integration Studio Transformations tree. When

you run the Clickstream Create Groups transformation, it creates a set of output views, one per group, based on the options specified on the **Options** tab. These output views are used as input to a later loop containing several calls to the Clickstream Sessionize transformation. In this way, the Clickstream Create Groups transformation bridges the output between first and second loops in a clickstream job.

See the Help for the **Options** tab for more information.



## Part 6

---

# Appendixes

<i>Appendix 1</i>	
<b>SAS Page Tag Predefined Data Elements Reference</b> .....	<a href="#">133</a>
<i>Appendix 2</i>	
<b>Clickstream Standard Columns Reference</b> .....	<a href="#">137</a>
<i>Appendix 3</i>	
<b>Server Software Reference</b> .....	<a href="#">145</a>





## Appendix 1

# SAS Page Tag Predefined Data Elements Reference

The following table lists the predefined data elements that can be collected by the SAS page tag, including the key name, value, and default enabled status:

**Table A1.1** Predefined Data Elements

Key Name	Description	Value	Default Enabled Status
VER	Displays the version number indicating the way the tag data is written.	2.2 (example)	Yes
EVT	Displays the type of user action or event that generated this line of data.	Valid values include load, click, and submit	Yes
RND	Indicates a random number.	Generated	Yes
CID	Displays the configurable ID that is included in a tag by default.	Default	Yes
VID	Displays the visitor ID that is created by storing a unique cookie in the user's browser. If cookies are enabled, this value is the same on each return visit to the Web site.	Generated	Yes
PID	Displays the page ID. This value is blank by default and serves as a place holder if a specific ID needs to be set when configuring the page code.	Not applicable	Yes
URI	Displays the Uniform Resource Indicator.	URI of Web page	Yes
REF	Displays the name of the referrer, which must be captured by the tag. In this context, the referrer is always the tagged page.	Not applicable	Yes
TTL	Displays the page title.	The title of the page	Yes
PROT	Displays the protocol of the URI being requested.	http or https	Yes
DOM	Displays the domain of the URI being requested.	W3C-domain	Yes

Key Name	Description	Value	Default Enabled Status
PORT	Displays the port of the URI being requested.	Port that received the request	Yes
CPU	Displays the CPU Class (when available).	x86 (example)	Yes
PLAT	Displays the platform (when available).	Win32 (example)	Yes
SINFO	Displays the screen resolution and color depth. Screen information is in the form of Width Height@Colors. For example, 1280x1024@32 indicates 1280 pixels wide by 1024 pixels high at 32-bit color depth.	1280x1024@32 (example)	Yes
FL	Displays whether Flash is enabled.	1 (true) or 0 (false)	Yes
FLV	Displays the Flash version.	WIN 10,0,22,87 (example)	Yes
CK	Displays whether cookies are enabled.	1 (true) or 0 (false)	Yes
JV	Displays whether Java is enabled.	1 (true) or 0 (false)	Yes
JVV	Displays the Java version.	1.5.0_11	Yes
JS	Displays whether JavaScript is enabled.	1 (true) or 0 (false)	Yes
SLNG	Displays the system language.	en-us (example)	Yes
BLNG	Displays the browser language.	en-us (example)	Yes
ULNG	Displays the user language.	en-us (example)	Yes
DT	Displays the client computer date.	4/7/2009 (example)	Yes
TM	Displays the client computer time.	16:1:48.663 (example)	Yes
M_Meta_Tag_Name	Displays the Meta tags.	Meta tag name/value pairs	Yes
C_Cookie_Name	Displays the Cookies tags.	Cookie tag name/value pairs	Yes
F_Form_Element_Name	Displays the Form Data (POST/GET) tags.	Form tag name/value pairs	No
CS	Contains the character set encoding of the data being collected. This setting is based on the character set encoding specified in the page or browser.	UTF-8	Yes

When available in the user's browser, the data elements listed in the preceding table are collected on each page load. Many are also collected (where applicable) by clicking a link or **Submit**.



## Appendix 2

# Clickstream Standard Columns Reference

### Overview

The clickstream transformations use various column definitions during their processing. Each of these columns is intended to fulfill a specific purpose during processing.

### Clickstream Parse Input Columns

The Clickstream Log transformation maps the columns from a Web log to the Clickstream Parse Input Columns and loads an output table with data from the log. This table becomes the input to the Clickstream Parse transformation.

The following table lists the metadata for the Clickstream Parse input columns.

**Table A2.1** Clickstream Log Output Columns

Column Name	Description	Label	Length	SAS Format
CLK_Client_IP	Specifies the visitor's IP address.	Client ID	64	\$64.
CLK_cs_Bytes	Specifies the number of bytes that the client sends to the server, upon a server request.	Bytes Received	8	COMMA15.
CLK_cs_Cookie	Specifies the raw cookie string.	Cookie String	4096	\$4096.
CLK_cs_Host	Specifies the host name, which is derived from the URL field that follows http://.	Requested Host	64	\$64.
CLK_cs_Method	Specifies the method that is used to submit the request (for example, POST or GET).	HTTP Method	16	\$16.
CLK_cs_Referrer	Specifies the full URL and any query parameters from the referring page.	Referrer	1332	\$1332.

Column Name	Description	Label	Length	SAS Format
CLK_cs_URI_Query	Specifies the query string that is passed to the URL.	Query Sting	1332	\$1332.
CLK_cs_URI_Stem	Specifies the URI, which is the URL, but without the http://www.domain.com/ field.	Requested File	1332	\$1332.
CLK_cs_UserAgent	Specifies the string that identifies the user's browser, which the user's browser sends.	User Agent	271	\$271.
CLK_cs_Username	Specifies the user name that the client used for authentication, if applicable.	Username	42	\$42.
CLK_cs_Version	Specifies the version of the HTTP protocol that is being used.	HTTP Version	8	\$8.
CLK_Date	Specifies the date stamp of the request.	Date	8	DATE9.
CLK_GMT_Offset	Specifies the Greenwich Mean Time (GMT) offset.	GMT Offset	5	\$5.
CLK_Null	Specifies the placeholder for a field that is not being used.	Null Variable	8	\$8.
CLK_s_Server	Specifies the server name, such as s-ComputerName.	Server Name	48	\$48.
CLK_s_Server_IP	Specifies the IP address of the Web server.	Server IP Address	16	\$16.
CLK_s_Server_Port	Specifies the number of the port that the Web server runs on.	Server Port	8	\$8.
CLK_s_Sitename	Specifies the name of the virtual Web site.	Site Name	32	\$32.
CLK_sc_Bytes	Specifies the number of bytes that the server sends to the client, upon a client request.	Bytes Sent	8	COMMA15.
CLK_sc_Status	Specifies the HTTP status code that the client receives from the server.	HTTP Status	8	4.

Column Name	Description	Label	Length	SAS Format
CLK_sc_SubStatus	Specifies the secondary status that is returned by some Web servers.	Sub Status	8	4.
CLK_Time	Specifies the timestamp of the request.	Time	8	TIME.
CLK_Time_Taken	Specifies the amount of time that is taken for the server to respond to the client request.	Time Taken	8	TIME.
CLK_sc_Win32_Status	Specifies the status that is returned by the Windows operating system.	Win32 Status	8	4.

### Clickstream Parse Output Columns

The Clickstream Parse transformation processes the columns received in its input table and produces an output table containing a derived set of output columns.

The following table lists the metadata for the Clickstream Parse output columns.

**Table A2.2** Clickstream Parse Output Columns

Column Name	Description	Completion Method	Label	Length	SAS Format
Browser	Specifies the type of browser that the visitor uses.	Is derived from CLK_cs_UserAgent, by using pattern matching on known browser names.	Browser	52	\$52.
Browser_Version	Specifies the version of the browser that the visitor uses.	Is derived from CLK_cs_UserAgent by using pattern matching to locate the browser name, and then extracting the version number that follows it.	Browser Version	16	\$16.
Bytes_Received	Specifies the number of bytes that the client sends to the server.	Pass-through CLK_cs_Bytes.	Bytes Received	8	COMMA15.
Bytes_Sent	Specifies the number of bytes that the server sends to the client.	Pass-through CLK_sc_Bytes.	Bytes Sent	8	COMMA15.

Column Name	Description	Completion Method	Label	Length	SAS Format
Client_IP	Specifies the visitor's IP address.	Pass-through CLK_Client_IP.	Client IP	64	\$64.
Cookie_Jar	Specifies the raw contents of the cookie jar.	Pass-through CLK_cs_Cookie.	Cookie Jar	4096	\$4096.
Date_Time	Specifies the date and time of the request.	Is derived by combining CLK_Date and CLK_Time.	Date and Time	8	DATETIME.
Domain	Specifies the host name.	Pass-through CLK_cs_Host.	Domain	165	\$165.
Method	Specifies the method that is used to submit the request (for example, POST or GET).	Pass-through CLK_cs_Method.	Method	16	\$16.
Platform	Specifies the hardware platform of the visitor's computer.	Is derived from CLK_cs_UserAgent, by using pattern matching on known platform names.	Platform	78	\$78.
Query_String	Contains the parameters that are specified in the URL. It is also referred to as the query or the CGI parameters.	Uses the pass-through CLK_URI_Query if non-blank. Otherwise, this query uses the query string from CLK_cs_URI_Stem.	Query String	1332	\$1332.
Record_ID	Specifies the unique identifier for each record.	Is derived by combining the date of the SAS process, the SAS process ID, and the record counter.	Record ID	24	\$24.
Referrer	Specifies the referring page (the URL from which the user requests access to the next URL).	Pass-through CLK_cs_Referrer.	Referrer	1332	\$1332.
Referrer_Domain	Specifies the domain of the referrer.	Is derived from CLK_cs_Referrer, and is the text that is located between the protocol (http://) and the first-level path (/).	Referrer Domain	165	\$165.



Column Name	Description	Completion Method	Label	Length	SAS Format
Referrer_Internal	Specifies whether the referrer is internal.	Is derived from a user-modified rule that runs after parse and sets referrer_internal to 1 when condition passes.	Referrer Internal	3	\$3.
Referrer_Query_String	Specifies the query string that is passed with the referrer.	Is derived from CLK_cs_Referrer, and is the text that is passed in the URL after the question mark (?).	Referrer Query String	1332	\$1332.
Referrer_Requested_File	Specifies the path and the filename of the referrer.	Is derived from CLK_cs_Referrer, and is all of the text that is located between the end of the domain name and the query string, if any.	Referrer Requested File	1332	\$1332.
Requested_File	Specifies the requested file.	Pass-through CLK_cs_URI_Stem.	Requested_File	1332	\$1332.
Server	Specifies the physical computer name that the Web server runs on, such as CLK_s_ComputerName.	Pass-through CLK_s_ComputerName.	Server	32	\$32.
Server_IP	Specifies the IP address of the Web server.	Pass-through CLK_s_IP.	Server IP Address	16	\$16.
Server_Port	Specifies the port that the Web server runs on, such as CLK_s_Port.	Pass-through CLK_s_Port.	Server Port	8	\$8.
Sitename	Specifies the name of the virtual Web site, such as CLK_s_SiteName.	Pass-through CLK_s_SiteName.	Site Name	48	\$48.
Status_Code	Specifies the HTTP status code that the server returns to the client during this request.	Pass-through CLK_sc_Status.	Status Code	8	4.

Column Name	Description	Completion Method	Label	Length	SAS Format
SubStatus	Specifies the secondary status that is returned by some Web servers.	Pass-through CLK_sc_SubStatus.	Sub Status	8	4.
User_Agent	Specifies the string that contains a description of the user's browser, which the user's browser sends.	Pass-through CLK_cs_UserAgent.	User Agent	271	\$271.
Username	Specifies the user name that the client sends to the server for authentication, if applicable.	Pass-through CLK_cs_Username.	Username	42	\$42.
Visitor_ID	Specifies a unique identifier for a visitor to the site. It typically contains the user's IP address and the name of the browser's user agent.	Is derived by combining CLK_Client_IP and CLK_cs_UserAgent, which is the default value, or by defining a user-defined rule that runs after the Clickstream Parse transformation.	Visitor Identifier	225	\$225.

### Clickstream Sessionize Output Columns

The Clickstream Sessionize transformation processes the columns received in its input table and produces an output table containing a derived set of output columns. Clickstream Sessionize also passes through each input column automatically.

The following table lists the metadata for the derived Clickstream Sessionize output columns.

Column Name	Description	Completion Method	Label	Length	SAS Format
Session_ID	The assigned session identifier for this visitor session.	User-Defined Rules or AutoFill e-Data Server configuration option (combines CLK_Client_IP and CLK_cs_UserAgent)	Session_ID	245	\$245
Session_Closed	Set to <b>1</b> to indicate that this record belongs to a closed session. Set to <b>0</b> to indicate that this record belongs to an open session.	Set to <b>1</b> when a session has exceeded the session timeout value. Otherwise, set to <b>0</b> .	Session_Closed	3	1.

Column Name	Description	Completion Method	Label	Length	SAS Format
Entry_Point	Set to <b>1</b> for the first click of the visitor's session. Set to <b>0</b> otherwise.	Clicks are examined in date_time order, and the first click entry_point is set to <b>1</b> . All others are set to <b>0</b> .	Entry_Point	3	
Exit_Point	Set to <b>1</b> for the last click of the visitor's session and it belongs to a closed session. Set to <b>2</b> for the last click of the visitor's session in an open session. Set to <b>0</b> otherwise. A value of <b>2</b> has been introduced to assist in the identification of exit points for open sessions.	Clicks are examined in date_time order, and final click exit_point is set to <b>1</b> . All others are set to <b>0</b> .	Exit_Point	3	1.
Eyeball_Time	Time the visitor has spent on the page before the next click.	Date_time of current click is subtracted from date_time of subsequent click. Last click in a session is set to missing.	Eyeball_Time	8	TIME.



## Appendix 3

# Server Software Reference

### Overview

This section details SAS Data Surveyor for Clickstream Data Server Software functionality that you can use during clickstream rules processing. You can also use it in other places within a clickstream job where custom code might be desired.

### ClickModifyParms

**Function:** ClickModifyParms( action, var, parmlist, delim1, delim2 );

**Purpose:** Modifies the parameters of a string by either dropping or keeping the specified clickstream parameters in the string. The edited string is returned. The following function is intended to be called from within a DATA step, on the right side of the = sign.

**Parameters:**

**action**

The action to take when a parmlist value is matched. Valid values are 'DROP' and 'KEEP' (with the quotes), indicating whether the values specified should be removed or kept, respectively. If action is not a recognized string, then the action defaults to 'DROP'. This value is not case sensitive.

**var**

The DATA step variable containing the parameters to parse.

**parmlist**

The space-delimited set of parameter names to search for in var. This value is not case sensitive.

**delim1**

The level 1 delimiter, which separates name value pairs. For example, & is standard in a query string.

**delim2**

The level 2 delimiter, which separates the name and value. For example, = is standard in a query string).

**Returns:** Returns the modified string.

Example:

```
* Example which drops s1, s3, and s4 parameters.;
* To keep only s1, s3, and s4 parameters, change DROP to KEEP in the call to;
* ClickModifyParms();
```

```
data _null_;
    attrib qs length=$300;
```

```
qs='s0=0&a=1&b=2&c=3&d=4&S1=1&S2=2&S3=3&e=5&S4=4&RTC=123&f=6&s100=5'  
thelist='s1 s3 s4';  
  
put "Original Query String:";  
put qs=;  
put "Parm List " thelist ":";  
qs=ClickModifyParms('DROP',qs,thelist, '&', '=');  
put "Modified Query String:";  
put qs=;  
  
run;
```