



THE  
POWER  
TO KNOW.

# **SAS<sup>®</sup> Data Surveyor for Clickstream Data 2.1**

## **User's Guide Second Edition**



The correct bibliographic citation for this manual is as follows: SAS Institute Inc. 2009. *SAS® Data Surveyor for Clickstream Data 2.1: User's Guide, Second Edition*. Cary, NC: SAS Institute Inc.

**SAS® Data Surveyor for Clickstream Data 2.1: User's Guide, Second Edition**

Copyright © 2009, SAS Institute Inc., Cary, NC, USA

ISBN 978-1-60764-385-2

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

**For a hard-copy 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 electronic book, November 2009

1st printing, November 2009

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

<b>Chapter 1 • Overview of SAS Data Surveyor for Clickstream Data</b>	<b>1</b>
How to Use This Document	1
What is SAS Data Surveyor for Clickstream Data?	2
Prerequisites	2
A Simple Clickstream Job	3
Clickstream Transformations	4
Clickstream Templates	5
Best Practices for Clickstream Jobs	7
Other Documentation	8
<b>Chapter 2 • Clickstream Log Transformation</b>	<b>9</b>
About the Clickstream Log Transformation	9
Specifying the Path to the Log	10
Maintaining Log Types	11
Managing User Columns	13
Specifying Log Options	14
<b>Chapter 3 • Clickstream Parse Transformation</b>	<b>15</b>
About the Clickstream Parse Transformation	16
Best Practices for the Clickstream Parse Transformation	17
Identifying Incoming Columns	18
Maintaining User Columns	19
Extracting Data from Clickstream Parameters	21
Applying Clickstream Parse Rules	22
Managing the Visitor ID	24
Managing Output Table Columns	25
Specifying Parse Options	25
<b>Chapter 4 • Clickstream Sessionize Transformation</b>	<b>27</b>
About the Clickstream Sessionize Transformation	27
Best Practices for the Clickstream Sessionize Transformation	30
Visitor ID Completion	30
Managing Non-Human Visitor Detection	31
Spanning Web Logs	32
Specifying Options for the Sessionize Transformation	33
<b>Chapter 5 • Basic Processing of a Clickstream Log</b>	<b>35</b>
About the Basic (Single) Web Log Template	35
Stages in the Single Log Template Job	36
Copying the Basic (Single) Web Log Template	38
Running a Single Log Job	39
<b>Chapter 6 • Processing Subsite Information</b>	<b>43</b>
About the Subsite Template Job	43
Stages in the Subsite Template Job	44
Copying the Sub Site Templates Folder	50
Managing Subsite Flow Segments	51
Running a Subsite Job	53
<b>Chapter 7 • Processing Multiple Clickstreams</b>	<b>57</b>
About the Basic (Multiple) Web Log Template Job	57

Best Practices for Multiple Log Jobs . . . . .	60
Stages in the Basic (Multiple) Web Log Template Job . . . . .	60
Copying the Basic (Multiple) Web Log Templates Folder . . . . .	70
Running a Multiple Logs Job . . . . .	71
<b>Chapter 8 • Processing Campaign Information . . . . .</b>	<b>75</b>
About the Customer Integration Template Job . . . . .	75
Stages in the Customer Integration Template Job . . . . .	76
Copying the Customer Integration Template Folder . . . . .	83
Collecting Campaign Information in a Customer Integration Job . . . . .	84
<b>Chapter 9 • Processing Tagged Pages . . . . .</b>	<b>91</b>
About Tagging Web Pages . . . . .	92
Best Practices for Page Tagging Jobs . . . . .	92
Preparing the Clickstream Collection Server . . . . .	93
Copying the Page Tagging Template . . . . .	93
JavaScript Page Tag Code . . . . .	94
Inserting a Minimal Tag . . . . .	95
Inserting a Full Page Tag . . . . .	96
Customizing a Full Page Tag . . . . .	98
Configuring Link Tracking in Tagged Pages . . . . .	104
Running a Page-Tagging ETL Job . . . . .	107
<b>Appendix 1 • Clickstream Parse Input and Output Columns . . . . .</b>	<b>111</b>
Clickstream Parse Input and Output Columns . . . . .	111
<b>Index . . . . .</b>	<b>117</b>

## Chapter 1

# Overview of SAS Data Surveyor for Clickstream Data

---

<b>How to Use This Document</b> . . . . .	<b>1</b>
<b>What is SAS Data Surveyor for Clickstream Data?</b> . . . . .	<b>2</b>
<b>Prerequisites</b> . . . . .	<b>2</b>
<b>A Simple Clickstream Job</b> . . . . .	<b>3</b>
<b>Clickstream Transformations</b> . . . . .	<b>4</b>
<b>Clickstream Templates</b> . . . . .	<b>5</b>
<b>Best Practices for Clickstream Jobs</b> . . . . .	<b>7</b>
Overview . . . . .	7
Backing Up Output Tables . . . . .	7
Resetting the CLICKRC Macro Variable . . . . .	7
<b>Other Documentation</b> . . . . .	<b>8</b>

---

## How to Use This Document

Suggestions for using this document are as follows:

- For an overview of the software, see [Chapter 1, “Overview of SAS Data Surveyor for Clickstream Data,”](#) on page 1.
- For a detailed introduction to the main transformations that are used in clickstream jobs, see [Chapter 2, “Clickstream Log Transformation ,”](#) on page 9, [Chapter 3, “Clickstream Parse Transformation,”](#) on page 15, and [Chapter 4, “Clickstream Sessionize Transformation,”](#) on page 27.
- For information about how clickstream transformations work together in the context of a job, see [Chapter 5, “Basic Processing of a Clickstream Log,”](#) on page 35.
- For information about specialized clickstream processing, see [Chapter 6, “Processing Subsite Information,”](#) on page 43, [Chapter 7, “Processing Multiple Clickstreams,”](#) on page 57, and [Chapter 9, “Processing Tagged Pages,”](#) on page 91.

## 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 online Web pages through various electronic devices. Clickstream data includes the stream of clicks stored in Web server logs. These clicks are generated by users as they browse Web sites.

The SAS Data Surveyor for Clickstream Data is a plug-in to SAS Data Integration Studio. This plug-in enables you to create jobs that extract and transform clickstream data from Web logs, and then 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 clickstream transformations, template jobs, and other components. The inputs to the jobs can be standard Web logs or enhanced logs that include clickstream data from tagged pages.

The SAS Data Surveyor for Clickstream Data enables you to:

- automate the extraction of useful information from large volumes of clickstream data
- use templates for common process flows that are used to cleanse and enrich clickstream data
- customize the template jobs for your own Web logs and outputs
- use 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

## Prerequisites

You must satisfy the following prerequisites in order to use the SAS Data Surveyor for Clickstream Data 2.1:

- All prerequisites for SAS Data Integration Studio 4.21 must be satisfied.
- Users 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

Components	Description	Where Components Are Installed
SERVER	<b>SAS Data Surveyor for Clickstream Data Server Components.</b> Includes macros and other server components that are required to execute clickstream jobs (jobs that include SAS Clickstream transformations).	On all SAS 9.2 Workspace Servers that execute clickstream jobs.

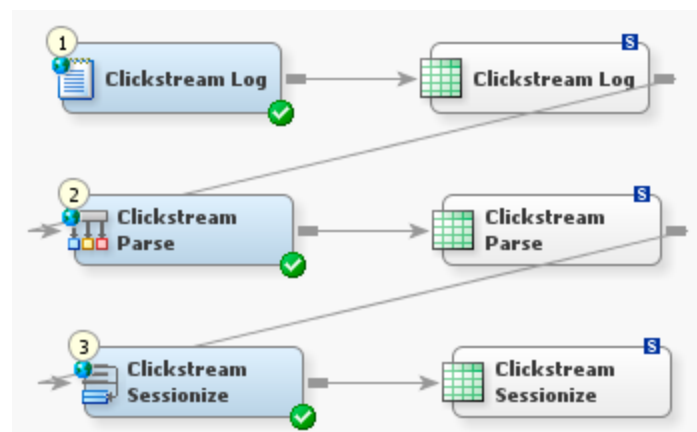
Components	Description	Where Components Are Installed
CLIENT	<b>SAS Data Surveyor for Clickstream Data Plug-ins.</b> Includes transformations, template jobs, and other components that are required to build clickstream jobs. For more information, see <a href="#">“Clickstream Transformations”</a> on page 4 and <a href="#">“Clickstream Templates”</a> on page 5.	On all computers where you want to use SAS Data Integration Studio to build clickstream jobs.
MID	<b>SAS Data Surveyor for Clickstream Data Mid-Tier.</b>  Updates an existing Apache Web server with Web content and configuration settings. These updates enable the Apache server to receive the output from SAS clickstream page tags, if these tags have been added to the Web pages that are being analyzed. The Apache Web server with the clickstream updates is called the clickstream collection server. For more information, see <a href="#">“Preparing the Clickstream Collection Server”</a> on page 93.	On a computer where an Apache HTTP Server has already been installed.

## A Simple Clickstream Job

The following display shows a simple clickstream job in SAS Data Integration Studio.

Only the main transformations that are provided with the SAS Data Surveyor for Clickstream Data are shown.

**Display 1.1** Simple Clickstream Job



In the simple job, information is read from a Web log, processed in various ways, and loaded into temporary work tables at the end of each step in the process flow. The following table describes the components in the job.

**Table 1.2** Transformations and Tables in the Simple Job

Name	Description	Inputs from and Outputs to
Clickstream Log transformation	Reads data from a clickstream log. Identifies the type of log to be processed. Maps input columns from the log to the <a href="#">“Clickstream Parse Input Columns” on page 111</a> . Loads an output table with data from the log. For more information, see <a href="#">Chapter 2, “Clickstream Log Transformation,” on page 9</a> .	From: Web log To: Log Output table
Clickstream Parse transformation	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="#">Chapter 3, “Clickstream Parse Transformation,” on page 15</a> .	From: Log Output table To: Parse Output table
Clickstream Sessionize transformation	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="#">Chapter 4, “Clickstream Sessionize Transformation,” on page 27</a> .	From: Parse Output table To: Sessionize Output table

In the clickstream jobs that are described in the rest of this document, some temporary work tables are replaced by permanent tables, checkpoint transformations are added to the flow, and other transformations are added to the flow. However, the main process flow for a clickstream job is similar to the flow for simple job in the preceding display.

---

## 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** folder. The Directory Contents transformation is added to the **Access** folder.

The main clickstream transformations are Clickstream Log, Clickstream Parse, and Clickstream Sessionize. For an overview of these transformations, see [“A Simple Clickstream Job” on page 3](#).



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

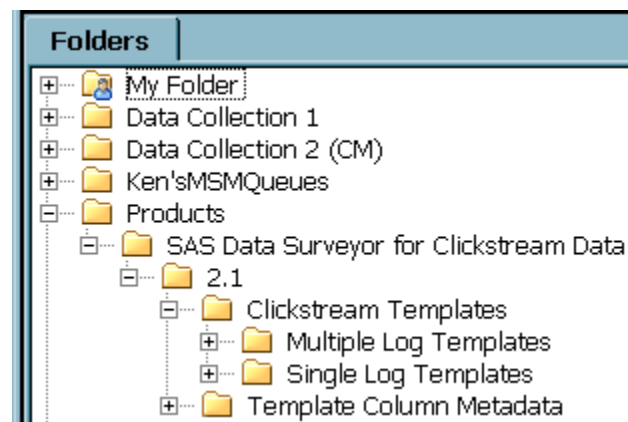
**Table 1.3** Specialized Clickstream Transformations

Name	Description
Clickstream Create Detail transformation	Combines the output from multiple Clickstream Sessionize transformations and creates a single data table. It is used in the Multiple Clickstream Log template job as described in <a href="#">“Create Detail and Generate Output” on page 69</a> .
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 Multiple Clickstream Log template job as described in <a href="#">“Combine Groups” on page 65</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 tests Web log data for the template jobs. Used in Clickstream Setup jobs.
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 Multiple Clickstream Log template job as described in <a href="#">“Prepare Data and Parameter Values to Pass to Loop 1” on page 61</a> .

## Clickstream Templates

The SAS Data Surveyor for Clickstream Data adds metadata for 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 and then the **SAS Data Surveyor for Clickstream Data** folder, as shown in the following display.

**Display 1.2** Clickstream Templates in the Products Folder



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 table describes the templates that are installed with the SAS Data Surveyor for Clickstream Data.

**Table 1.4** Clickstream Templates

Name	Description
Basic (Multiple) Web Log Template	<p>Enables you to process multiple clickstream logs from multiple servers.</p> <p>Includes a setup job (<b>clk_0010_setup_basic_multi_job</b>), the job template (<b>clk_0020_load_multi_dd</b>s), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">Chapter 7</a>, “Processing Multiple Clickstreams,” on page 57.</p>
Customer Integration Template	<p>Enables you to capture 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 (<b>clk_0010_setup_basic_ci_job</b>), the job template (<b>clk_0020_load_ci_dd</b>s), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see “Processing Campaign Information” on page 75.</p>
Basic (Single) Web Log Template	<p>Enables you to process a single clickstream log.</p> <p>Includes a setup job (<b>clk_0010_setup_basic</b>), the job template (<b>clk_0020_create_output_detail</b>), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">Chapter 5</a>, “Basic Processing of a Clickstream Log,” on page 35.</p>
Page Tagging Template	<p>Enables you to process a clickstream log that includes page tagging data.</p> <p>Includes a setup job (<b>clk_0010_setup_page_tagging</b>), the job template (<b>clk_0020_page_tagging_detail</b>), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">Chapter 9</a>, “Processing Tagged Pages,” on page 91.</p>
Subsite Template	<p>Enables you to process a 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 (<b>clk_0010_setup_sub_site</b>), the job template (<b>clk_0020_create_sub_site_tables</b>), and metadata objects for sample data under the <b>Data Sources</b> folder. For more information about this template, see <a href="#">Chapter 6</a>, “Processing Subsite Information,” on page 43.</p>
Template Column Metadata	<p>Provides a repository of column definitions that are useful in clickstream jobs.</p> <p>Includes the metadata for a number of tables and columns that are used in clickstream jobs.</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 should be able to run the template jobs to verify that all servers and other components are working properly.

## Best Practices for Clickstream Jobs

### Overview

The following best practices apply to clickstream jobs in general.

### Backing Up Output Tables

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 1.5** 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.	Back up the data output table for each Sessionize transformation.  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.
Temporary work tables for parameters and rules that are output from the Parse transformation.	Redirect the temporary work tables for parameters and rules to a permanent library. Then back up this permanent library.  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> .
Temporary work tables for spiders and sessions that are output from the Sessionize transformation.	Redirect the temporary work tables for spiders and sessions to a permanent library. Then back up this permanent library.  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> .

### 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 nonzero 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 session.
- 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;`
- Open the properties window for the affected transformation, click the **Precode and Postcode** tab, and enter the following code in the **Precode** window: `%LET CLICKRC=0;`

---

## Other Documentation

For more information about the page tagging API that is described in [Chapter 9, “Processing Tagged Pages,”](#) on page 91, see the *SAS Data Surveyor for Clickstream Data 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

For more information about SAS Data Integration Studio, see the *SAS Data Integration Studio: User's Guide* at <http://support.sas.com/documentation/onlinedoc/etls/>.

## Chapter 2

# Clickstream Log Transformation

---

<b>About the Clickstream Log Transformation</b> . . . . .	<b>9</b>
<b>Specifying the Path to the Log</b> . . . . .	<b>10</b>
Problem . . . . .	10
Solution . . . . .	10
Tasks . . . . .	11
<b>Maintaining Log Types</b> . . . . .	<b>11</b>
Problem . . . . .	11
Solution . . . . .	11
Tasks . . . . .	12
<b>Managing User Columns</b> . . . . .	<b>13</b>
Problem . . . . .	13
Solution . . . . .	13
Tasks . . . . .	13
<b>Specifying Log Options</b> . . . . .	<b>14</b>

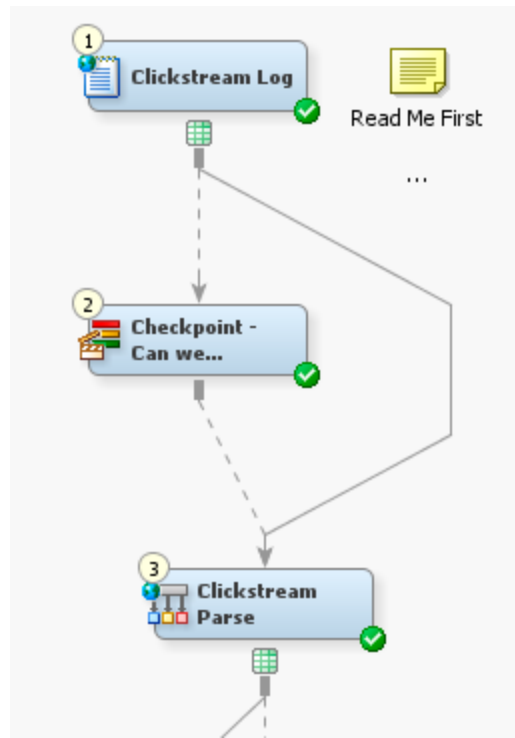
---

## 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 template jobs that are provided with the SAS Data Surveyor for Clickstream Data.

**Display 2.1** Clickstream Log Transformation in the Basic (Single) Web Log Job



Typical user tasks for the Clickstream Log transformation include:

- 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

### Problem

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.

It is assumed that you are working with a copy of one of the template jobs that are described in [“Clickstream Templates” on page 5](#), or that you have dragged and dropped the Clickstream Log transformation into another job.

### Solution

Use the **File Location** tab in the properties window for the Clickstream Log transformation to specify the location of the Web log.

## Tasks

### 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 are specifying 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.

---

## Maintaining Log Types

### Problem

You want to view, add, or update the log type definitions that are used in the Clickstream Log transformation.

### Solution

You can use the **Log Types** tab in the properties window for the Clickstream Log transformation. 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 (page tagging 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.

## Tasks

### 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 such that your new log type is recognized first, or disable the log types that you no longer want to process in this job.

#### 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

### Problem

You have an input column from the Web log that does not have a matching Clickstream Parse Input Column.

### Solution

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 13](#)
- [“Reuse User-Defined Columns in Other Clickstream Jobs” on page 14](#)
- [“Other Tasks for Managing User Columns” on page 14](#)

### Tasks

#### 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 3

# Clickstream Parse Transformation

---

<b>About the Clickstream Parse Transformation</b> . . . . .	<b>16</b>
<b>Best Practices for the Clickstream Parse Transformation</b> . . . . .	<b>17</b>
Handling Non-Human Visitors in the Clickstream Parse Transformation . . . . .	17
Maintaining the Hold Buffer Size Setting . . . . .	17
Optimizing Sort Using SORTSIZE . . . . .	17
Setting a Visitor ID Value . . . . .	18
<b>Identifying Incoming Columns</b> . . . . .	<b>18</b>
Problem . . . . .	18
Solution . . . . .	18
Tasks . . . . .	19
<b>Maintaining User Columns</b> . . . . .	<b>19</b>
Problem . . . . .	19
Solution . . . . .	19
Tasks . . . . .	20
<b>Extracting Data from Clickstream Parameters</b> . . . . .	<b>21</b>
Problem . . . . .	21
Solution . . . . .	21
Tasks . . . . .	21
<b>Applying Clickstream Parse Rules</b> . . . . .	<b>22</b>
Problem . . . . .	22
Solution . . . . .	22
Tasks . . . . .	23
<b>Managing the Visitor ID</b> . . . . .	<b>24</b>
Problem . . . . .	24
Solution . . . . .	24
Tasks . . . . .	24
<b>Managing Output Table Columns</b> . . . . .	<b>25</b>
Problem . . . . .	25
Solution . . . . .	25
Tasks . . . . .	25
<b>Specifying Parse Options</b> . . . . .	<b>25</b>

## About the Clickstream Parse Transformation

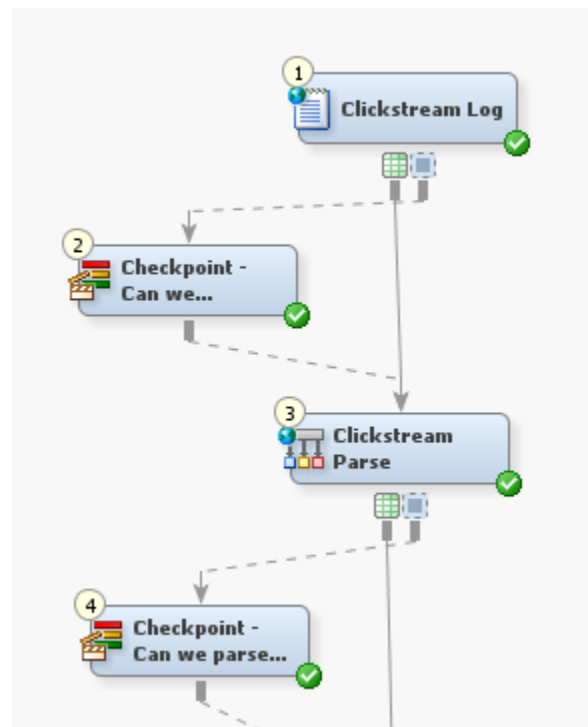
The Clickstream Parse transformation usually reads the output table from the Clickstream Log transformation. However, the Clickstream Parse transformation can read from any input table. Then, it interprets this incoming data and creates a common set of output columns, independent of the incoming Web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

The Clickstream Parse transformation performs the following functions:

- associates input columns with “Clickstream Parse Input Columns” on page 111 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
- uses built-in rules to determine values for Browser Type, Browser Version, and Platform

The Clickstream Parse transformation is shown in the following display.

**Display 3.1** 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 31](#).

### *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 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) and 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.

---

## Identifying Incoming Columns

### Problem

You want to identify the meaning of the incoming columns to the Clickstream Parse transformation. To do this, 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 111](#).

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). Additionally, 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.

### Solution

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.

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 19](#)
- [“Other Tasks for Input Mapping” on page 19](#)

## Tasks

### **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

### **Problem**

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.

### **Solution**

You can use the **User Columns** tab in the properties window for the Clickstream Parse transformation. Each row in the table on the **User Columns** 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 20](#)
- [“Reuse User-Defined Columns in Other Clickstream Jobs” on page 20](#)
- [“Other Tasks for Managing User Columns” on page 20](#)

## Tasks

### **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.



---

## Extracting Data from Clickstream Parameters

### Problem

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

### Solution

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 Parameter” on page 21](#)
- [“Other Tasks for Managing Parameters” on page 21](#)

### Tasks

#### **Create a New Parameter**

Perform the following steps to create a new parameter:

1. Click **Create a new parameter** to add a row to the parameters table.
2. 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.
3. Select a source type from the drop-down menu in the Source Type column.
4. 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 **C:**

`\Documents and Settings\<user ID>` 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**.
- Consider creating or importing any user columns 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.

---

## Applying Clickstream Parse Rules

### Problem

You want to perform specified record-level processes that are based on the content of the clickstream input data that is 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

### Solution

You can use the **Rules** tab in the properties window for the Clickstream Parse transformation. 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, to be applied 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, thus saving 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 23](#)
- [“Other Tasks for Managing Rules” on page 23](#)

## Tasks

### 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. For information about modifying an existing rule, see the steps that discuss the **Rules** tab in [“Adding Subsite Flow Segments” on page 51](#).

If you create many customized rules, it is good practice to 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 **C:\Documents and Settings\<user ID>** 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

### Problem

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.

### Solution

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. 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 first 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.

### Tasks

#### **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 first 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

### Problem

You want to select the columns that are present in the output from the Clickstream Parse transformation.

### Solution

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, you can select the column from this list in the **Available columns** list box and 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.

### Tasks

#### ***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 17.](#))

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

## Chapter 4

# Clickstream Sessionize Transformation

---

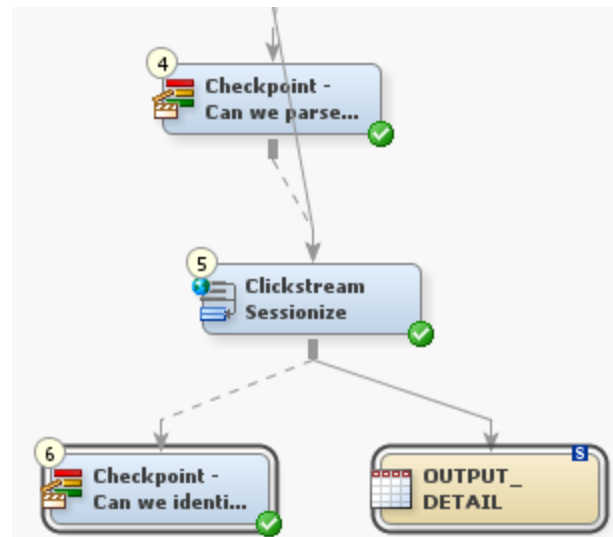
<b>About the Clickstream Sessionize Transformation</b> . . . . .	<b>27</b>
<b>Best Practices for the Clickstream Sessionize Transformation</b> . . . . .	<b>30</b>
Backing Up PERMLIB . . . . .	30
Managing the Contents of PERMLIB . . . . .	30
<b>Visitor ID Completion</b> . . . . .	<b>30</b>
Overview . . . . .	30
Process . . . . .	31
<b>Managing Non-Human Visitor Detection</b> . . . . .	<b>31</b>
Overview of Non-Human Visitors . . . . .	31
Problem . . . . .	32
Solution . . . . .	32
Task . . . . .	32
<b>Spanning Web Logs</b> . . . . .	<b>32</b>
<b>Specifying Options for the Sessionize Transformation</b> . . . . .	<b>33</b>
Input Options . . . . .	33
Tables Options . . . . .	33
Tuning Options . . . . .	33

---

## 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 4.1** Clickstream Sessionize Transformation

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

**Table 4.1** 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 (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 that 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 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 they were in before the last execution. Otherwise, duplicate data appears in the output table for the Clickstream Sessionize transformation.

An easy way to remove the tables contained in PERMLIB is to add code similar to the following in the **Precode** panel on the **Precode and Postcode** tab:

```
libname permlib '<your PERMLIB path goes here>';  
proc datasets lib=permlib kill nowarn nolist;  
run;
```

Remove this pre-code when you are ready to run the job in production so that session data is correctly connected across executions of the job. The nolist option prevents a job warning if the PERMLIB directory is empty when you execute this code.

---

## 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, it is possible to analyze this visitor activity to determine the original referring site from which the visitor came. For example, this can be useful in

determining how much revenue (during cart checkout) 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, but 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.

---

# Managing Non-Human Visitor Detection

## Overview of Non-Human Visitors

Spiders, robots, crawlers, pingers, and any other computer program 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 such as 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 NHV activity is handled is during the Clickstream Sessionize transformation, using 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 **Options** tab help for details on how to configure this functionality.

**Problem**

You have already filtered and removed the NHVs found by the Clickstream Parse transformation using the rule that examines the User Agent string, but you want to analyze the visitor behavior to ensure that none of the remaining sessions were created by NHVs.

**Solution**

Set the options in the Clickstream Sessionize properties window to detect any NHVs.

**Task**

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 and allows you to validate that the detection heuristics are accurate. 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.

---

## Spanning Web Logs

If a session extends from one Web log into 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 30](#).

## 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.

### 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:

- **Additional output library** stores output data including records that are considered non-human interactions such as 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.)
- **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 use 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 5

# Basic Processing of a Clickstream Log

---

<b>About the Basic (Single) Web Log Template</b> . . . . .	<b>35</b>
<b>Stages in the Single Log Template Job</b> . . . . .	<b>36</b>
Overview . . . . .	36
Load and Prepare Clickstream Log Data . . . . .	36
Parse Data . . . . .	37
Create Sessions and Generate Output . . . . .	37
<b>Copying the Basic (Single) Web Log Template</b> . . . . .	<b>38</b>
<b>Running a Single Log Job</b> . . . . .	<b>39</b>
Problem . . . . .	39
Solution . . . . .	40
Tasks . . . . .	40

---

## About the Basic (Single) Web Log Template

The basic (single) Web log template enables you to process only one clickstream log in a job. This template is useful when you need to do a trial run on a single log to determine whether all of the data in a clickstream log can be read properly. You can also use it whenever you need to process a single clickstream log, regardless of the log file size. In situations that require you to process multiple larger logs or enable you to split a single very large log, the multiple Web log template yields better performance because it uses parallel processing to process multiple logs at the same time. For more information, see [“About the Basic \(Multiple\) Web Log Template Job” on page 57](#).

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

## Stages in the Single Log Template Job

### Overview

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

- “Load and Prepare Clickstream Log Data” on page 36
- “Parse Data” on page 37
- “Create Sessions and Generate Output” on page 37

### 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 single log template process uses a Clickstream Log transformation to locate the clickstream log data and prepare a SAS data table or view that can be processed further.

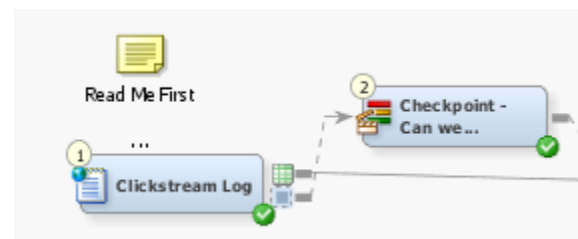
The transformations in this stage are described in the following table:

**Table 5.1** Load and Prepare Clickstream Log Data Transformations

Name	Description	Inputs from and Outputs to
Clickstream Log transformation	Extracts the raw Web log data and creates a SAS data table or view.	From: None To: Clickstream Parse transformation; Checkpoint - Can we recognize the log ? transformation
Checkpoint - Can we recognize the 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 template job that runs this stage:

**Display 5.1** Load and Prepare Stage Process Flow





## Parse Data

The second stage of the single log template process uses a Clickstream Parse transformation to parse the data and create meaningful columns.

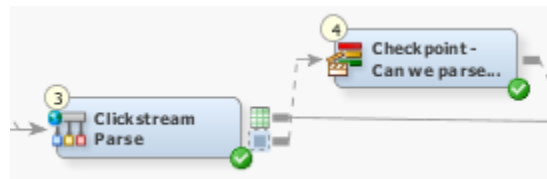
The transformations in this stage are described in the following table:

**Table 5.2** 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; Checkpoint - Can we recognize the log? transformation  To: Checkpoint - Can we parse the log? transformation; Clickstream Sessionize transformation
Checkpoint - Can we parse the log? 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 template job that runs this stage:

**Display 5.2** Parse Data Stage Process Flow



## Create Sessions and Generate Output

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

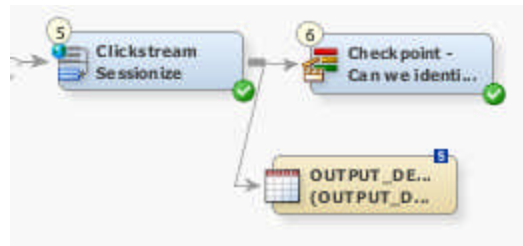
The transformations in this stage are described in the following table:

**Table 5.3** Create Sessions and Generate Output Transformations and Tables

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; Checkpoint - Can we parse the log? transformation To: Checkpoint - Can we identify sessions? transformation; OUTPUT_DETAIL table
Checkpoint - Can we identify sessions? 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: None
OUTPUT_DETAIL table	Contains the output from the processed clickstream log file.	From: Clickstream Sessionize transformation To: None

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

**Display 5.3** Sessions and Output Stage Process Flow



## Copying the Basic (Single) Web Log Template

You should copy the Basic Web Log Templates folder located under the Single Log Templates folder before you modify any of the objects 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.

Perform the following steps to copy and prepare the single log template:

1. Right-click the **Basic Web Log Templates** folder. Then, click **Copy** in the pop-up menu.
2. Right-click the folder where you want to paste the template. Then, click **Paste Special** in the pop-up menu 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:* 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 **Basic Web Log Templates** folder that was just copied. Then, open the properties windows for the two jobs in the 2.1 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.
4. Expand the Data Sources folder for the template and its subfolders to reveal the libraries used by the single log job. To distinguish these libraries from the original libraries used by the Page Tagging 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.
5. If you modified the Directory Paths when copying the multiple log 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 DirectoryName** 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.
6. 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.
7. 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.
8. 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.

---

## Running a Single Log Job

### Problem

You want to process a single clickstream log.

## Solution

You can process the job in the single log job template. If you have not done so already, you should run a copy of the setup job for the single log template, which is named `clk_0010_setup_basic`. When you actually process the data, you should run a copy of the single log job, which is named `clk_0200_create_output_details`. By running a copy, you protect the original template. For information about running the setup job and creating a copy of the original job, see [“Copying the Sub Site Templates Folder” on page 50](#).

## Tasks

### Review and Prepare the Job

You can examine the single log 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.

Perform the following steps to review and prepare the job:

1. Open the renamed single log 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.

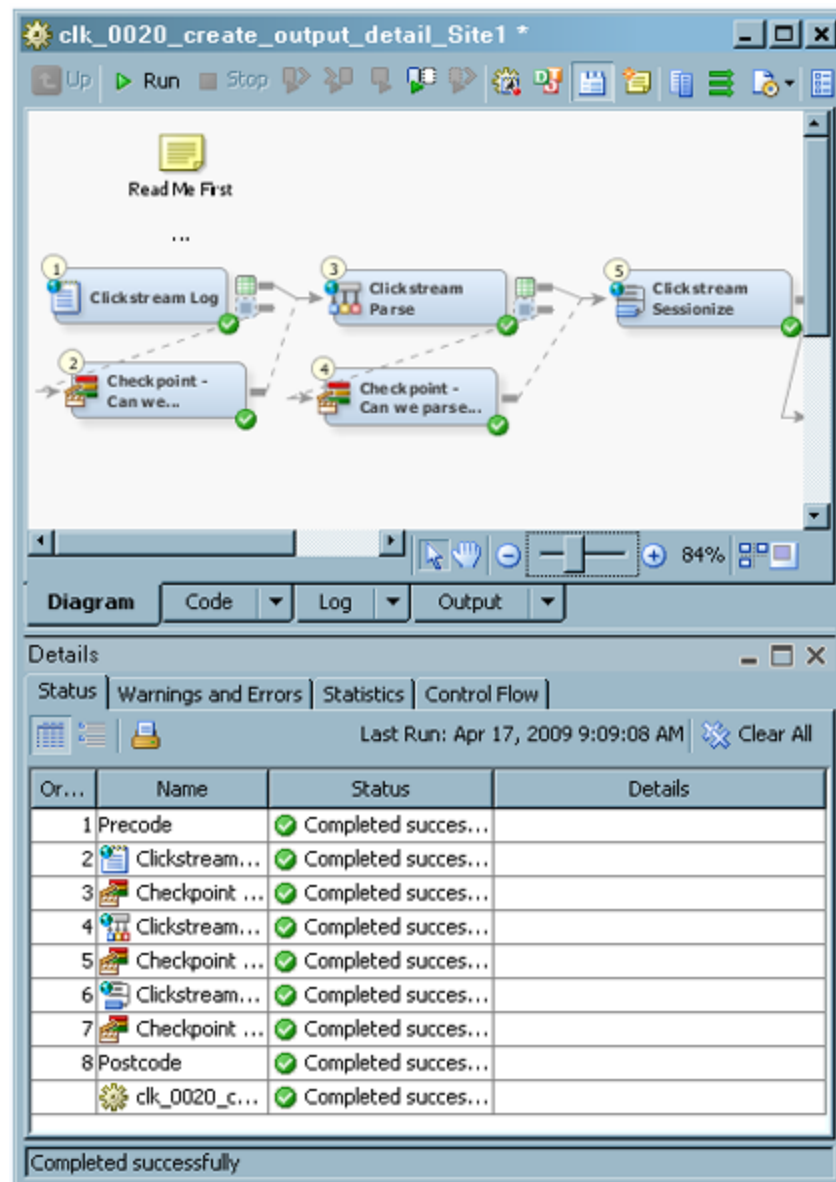
### Run the Job and Examine the Output

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

1. Run the job.

The following display shows a successfully completed sample job.

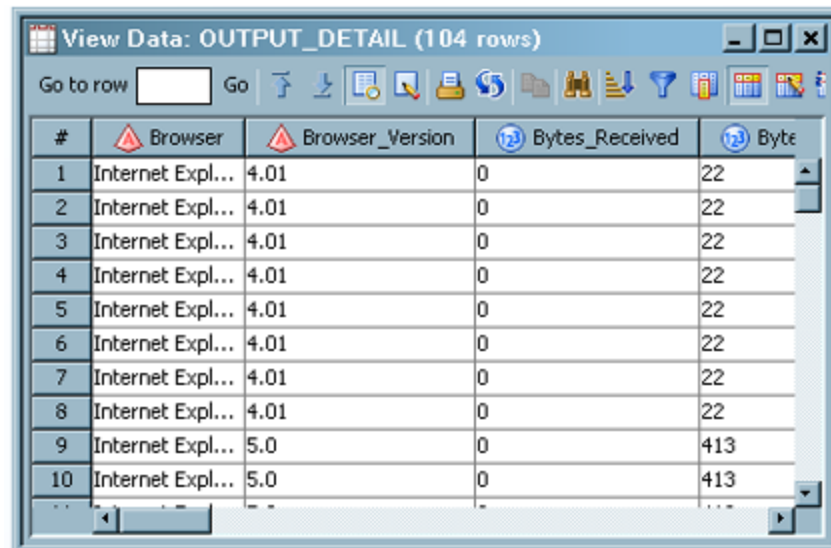
**Display 5.4** Completed Single Log Job



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

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

**Display 5.5** Single Log Job Output



The screenshot shows a software window titled "View Data: OUTPUT\_DETAIL (104 rows)". It contains a table with 104 rows and 4 columns. The columns are labeled "#", "Browser", "Browser\_Version", "Bytes\_Received", and "Byte". The first 10 rows are visible, showing data for Internet Explorer 4.01 and 5.0. The "Bytes\_Received" column shows 0 for rows 1-8 and 9-10, and 413 for rows 9-10. The "Byte" column shows 22 for rows 1-8 and 413 for rows 9-10.

#	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

## Chapter 6

# Processing Subsite Information

---

<b>About the Subsite Template Job</b> .....	<b>43</b>
<b>Stages in the Subsite Template Job</b> .....	<b>44</b>
Overview .....	44
Load Data and Apply Global Rules .....	44
Generate Subsite Sessions .....	45
Generate Data from Site-Wide Data .....	48
<b>Copying the Sub Site Templates Folder</b> .....	<b>50</b>
<b>Managing Subsite Flow Segments</b> .....	<b>51</b>
Problem .....	51
Solution .....	51
Tasks .....	51
<b>Running a Subsite Job</b> .....	<b>53</b>
Problem .....	53
Solution .....	54
Tasks .....	54

---

## About the Subsite Template Job

The subsite template job enables you to identify one or more subsites within a Web log. Then, you can identify and session the data for only those subsites that you need to analyze. All other data in the Web log is filtered out.

Subsites are commonly identified in the following ways:

- subsite specification via URI: `http://www.abc.com/marketing` and `http://www.abc.com/techsupp`, where the organization identifies the subsites with part of the URI (marketing and techsupp in these URIs)
- subsite specification via sub domains: `http://mkt.abc.com` and `http://ts.abc.com`, where mkt and ts are considered sub-domains of the abc.com domain
- subsites to be defined by the user, where the user identifies subsites by using some user-defined algorithm against the Web log data (such as the information stored in a cookie string)

The subsite job uses the following Clickstream Parse transformations that are configured for specialized purposes and renamed accordingly:

- Clickstream Parse - Global Rules, which filters out superfluous data such as graphic files, non-pages, and spiders that identify themselves in their user agent string
- Clickstream Parse - Subsite, which isolates subsites
- Clickstream Parse - ALL, which generates output that includes the content from all subsites

The subsite job also includes a series of Clickstream Sessionize transformations to split the sets of data into sessions.

## Stages in the Subsite Template Job

### Overview

The Subsite Template Job can be divided into the following stages:

- [“Load Data and Apply Global Rules” on page 44](#)
- [“Generate Subsite Sessions” on page 45](#)
- [“Generate Data from Site-Wide Data” on page 48](#)

### Load Data and Apply Global Rules

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 subsite template process locates the data and applies global rules to it. For example, you can apply default rules to filter requests for image files or requests made by spiders and robots that identify themselves as such in their user agent data. For more information, see [“Managing Non-Human Visitor Detection” on page 31](#).

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

**Table 6.1** Load Data and Apply Global Rules Transformations

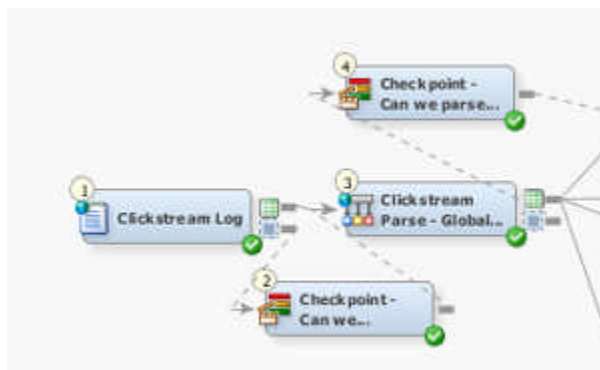
Name	Description	Inputs from and Outputs to
Clickstream Log transformation	Extracts data from the Web log in the specified file location.	From: Specified file location for Web log  To: Checkpoint - Can we recognize the log? transformation; Clickstream Parse - Global Rules transformation
Checkpoint - Can we recognize the 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 - Global Rules transformation



Name	Description	Inputs from and Outputs to
Clickstream Parse - Global Rules transformation	Parses the data and applies global rules that apply to all of the subsites; filters out graphics files, non-pages, and spiders that identify themselves in their user agent strings. Also see “ <a href="#">Managing Non-Human Visitor Detection</a> ” on page 31.	From: Clickstream Log transformation; Checkpoint - Can we recognize the log? transformation  To: Checkpoint - Can we parse the log? transformation; Clickstream Parse - PRD transformation; Clickstream Parse - SVCS transformation; Clickstream Parse - GEN transformation; Clickstream Parse - ALL transformation
Checkpoint - Can we parse the log? transformation	Evaluates the return code from Clickstream Parse - Global Rules; sends e-mail to specified address if the parse step fails.	From: Clickstream Parse - Global Rules transformation  To: Clickstream Parse - PRD transformation

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

**Display 6.1** Global Rules Stage Process Flow



## Generate Subsite Sessions

The second stage of the subsite template process uses a Clickstream Parse transformation to limit the data to a selected subsite. Then, a Clickstream Sessionize transformation is used to identify the sessions in that particular subsite. You can assign a session ID, which effectively identifies the sessions that are present within the data.

The template job performs this operation for three distinct subsites: PRD, SVCS, and GEN. Of course, you do not have to process exactly this set of subsites. The template is meant to serve only as an example. You can filter the data for as many or as few subsites as needed. Simply add or remove sets of transformations to match the number of subsites that you have. Then, change the names to appropriate values.

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

**Table 6.2** *Generate Subsite Sessions Transformations and Tables*

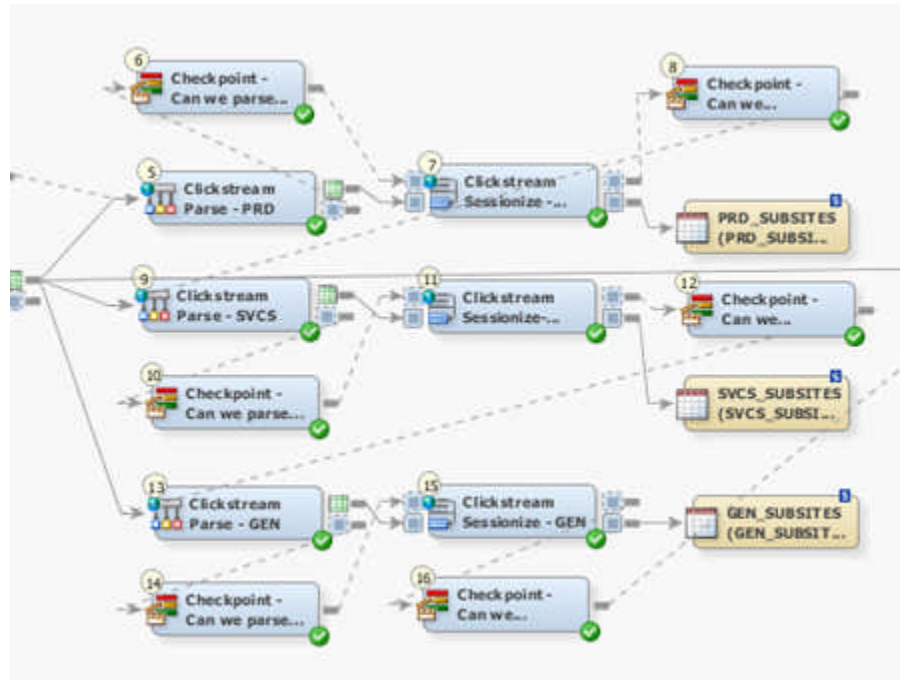
Name	Description	Inputs from and Outputs to
PRD subsite		
Clickstream Parse - PRD transformation	Parses the data for the PRD subsite; all other data is filtered out.	From: Clickstream Parse - Global Rules; Checkpoint - Can we parse the log? To: Checkpoint - Can we parse PRD Subsite data? transformation; Clickstream Sessionize - PRD transformation
Checkpoint - Can we parse PRD Subsite data? transformation	Evaluates the return code from Clickstream Parse - PRD; sends e-mail to specified address if the parse step fails.	From: Clickstream Parse - PRD transformation To: Clickstream Sessionize - PRD transformation
Clickstream Sessionize - PRD transformation	Identifies sessions within PRD subsite data.	From: Checkpoint - Can we parse PRD Subsite data? transformation; Clickstream Parse - PRD transformation To: Checkpoint - Can we sessionize PRD Subsite data? transformation; PRD_SUBSITES table
Checkpoint - Can we sessionize PRD Subsite data? transformation	Evaluates the return code from Clickstream Sessionize - PRD; sends e-mail to specified address if the sessionize step fails.	From: Clickstream Sessionize - PRD transformation To: Clickstream Parse - SVCS transformation
PRD_SUBSITES table	Contains the output from the PRD subsite.	From: Clickstream Sessionize - PRD transformation To: None
SVCS subsite		
Clickstream Parse - SVCS transformation	Parses the data for the SVCS subsite; all other data filtered out.	From: Clickstream Parse - Global Rules; Checkpoint - Can we sessionize PRD Subsite data? transformation To: Checkpoint - Can we parse the SVCS Subsite data? transformation

Name	Description	Inputs from and Outputs to
Checkpoint - Can we parse the SVCS Subsite data? transformation	Evaluates the return code from Clickstream Parse - SVCS; sends e-mail to specified address if the parse step fails.	From: Clickstream Parse - SVCS transformation To: Clickstream Sessionize - SVCS transformation
Clickstream Sessionize - SVCS transformation	Identifies sessions within SVCS subsite data.	From: Checkpoint - Can we parse the SVCS Subsite data? transformation To: Checkpoint - Can we sessionize SVCS subsite data?; SVCS_SUBSITES table
Checkpoint - Can we sessionize SVCS subsite data? transformation	Evaluates the return code from Clickstream Sessionize - SVCS; sends e-mail to specified address if the sessionize step fails.	From: Clickstream Sessionize - SVCS transformation To: Clickstream Parse - GEN transformation
SVCS_SUBSITES table	Contains the output from the SVCS subsite.	From: Clickstream Sessionize - SVCS transformation To: None
GEN subsite		
Clickstream Parse - GEN transformation	Parses the data for the GEN subsite; all other data filtered out.	From: Clickstream Parse - Global Rules transformation; Checkpoint - Can we sessionize SVCS subsite data? transformation To: Checkpoint - Can we parse GEN subsite data? transformation
Checkpoint - Can we parse GEN subsite data? transformation	Evaluates the return code from Clickstream Parse - GEN; sends e-mail to specified address if the parse step fails.	From: Clickstream Parse - GEN transformation To: Clickstream Sessionize - GEN transformation
Clickstream Sessionize - GEN transformation	Identifies sessions within GEN subsite data.	From: Checkpoint - Can we parse GEN subsite data? transformation To: Checkpoint - Can we sessionize GEN subsite data? transformation; GEN_SUBSITES table
Checkpoint - Can we sessionize GEN subsite data? transformation	Evaluates the return code from Clickstream Sessionize - GEN; sends e-mail to specified address if the sessionize step fails.	From: Clickstream Sessionize - GEN transformation To: Clickstream Parse - ALL transformation

Name	Description	Inputs from and Outputs to
GEN_SUBSITES table	Contains the output from the GEN subsite.	From: Clickstream Sessionize - GEN transformation To: None

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

**Display 6.2** Subsites Stage Process Flow



### Generate Data from Site-Wide Data

The third stage of the subsite template processes the data from the Web log without splitting it into subsites. This stage enables you to create an output table that covers all the data in the Web log. Although filters are not applied in this data, this data can be thought of as a subsite of everything. For example, the ALL output data might be of interest to those responsible for the entire company's site, while the PRD data might be of interest to those in charge of the PRD department's site.

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

**Table 6.3** Generate Data from Site-Wide Data Transformations and Tables

Name	Description	Inputs from and Outputs to
Clickstream Parse - ALL transformation	Parses the data for the entire Web log; no subsite data filtered out.	From: Clickstream Parse - Global Rules transformation; Checkpoint - Can we sessionize GEN subsite data? transformation To: Checkpoint - Can we parse ALL Subsite data? transformation
Checkpoint - Can we parse ALL Subsite data? transformation	Evaluates the return code from Clickstream Parse - ALL; sends e-mail to specified address if the parse step fails.	From: Clickstream Parse - ALL transformation To: Clickstream Sessionize - ALL transformation
Clickstream Sessionize - ALL transformation	Identifies sessions from the undivided Web log into sessions.	From: Checkpoint - Can we parse ALL Subsite data? transformation To: Checkpoint - Can we sessionize ALL subsites? transformation; ALL_SUBSITES table
Checkpoint - Can we sessionize ALL subsites? transformation	Evaluates the return code from Clickstream Sessionize - ALL; sends e-mail to specified address if the sessionize step fails.	From: Clickstream Sessionize - ALL transformation To: None
ALL_SUBSITES table	Contains the output that has not been divided into subsites.	From: Clickstream Sessionize - ALL transformation To: None

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

**Display 6.3** Site-Wide Data Stage Flow



## Copying the Sub Site Templates Folder

You should copy the Sub Site Templates folder before you modify any of the objects 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.

Perform the following steps to copy and prepare the subsite template:

1. Right-click the Sub Site Templates folder. Then, click **Copy** in the pop-up menu.
2. Right-click the folder where you want to paste the template. Then, click **Paste Special** in the pop-up menu 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:* 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 for objects such as SAS libraries. Click **Finish** when you complete the pages.

3. Rename (if desired) and expand the new Sub Site Templates folder that was just copied. Then, open the properties window for the two jobs in the 2.1 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\_sub\_site job to clk\_0010\_setup\_sub\_site\_Site1 and the clk\_0020\_sub\_site\_tables job to clk\_0020\_sub\_site\_tables\_Site1.
4. If you modified the directory paths when copying the multiple log templates, then open the renamed clk\_0010\_setup\_sub\_site job and modify the Setup transformation properties. If you modified the Directory Paths when copying the Sub Site Template, then open the properties on the renamed clk\_0010\_setup\_sub\_site job. (Otherwise, proceed to step 5.) Then, on the **Options** tab, modify the values in the **Root Directory** and **Template DirectoryName** 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\_sub\_site job. This job creates the necessary folders and sample data to support the renamed clk\_0020\_sub\_site\_tables job.
6. Open the job properties window in the renamed clk\_0020\_sub\_site\_tables 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. Open the properties window for the Clickstream Log transformation and specify the appropriate value in the **File name** field on the **File Location** tab.

---

## Managing Subsite Flow Segments

### Problem

The subsite template job contains transformations that isolate three separate subsites from the clickstream log. Of course, your data often includes a larger or smaller number of sites. Even if you need to process exactly three subsites, they are unlikely to be named PRD, SVCS, and GEN. Fortunately, you can add, delete, and modify the subsite flow segments in your jobs.

### Solution

You can manage your subsite flow segments in the following ways:

- [“Adding Subsite Flow Segments” on page 51](#)
- [“Deleting Existing Subsite Flow Segments” on page 53](#)
- [“Modifying Existing Subsite Flow Segments” on page 53](#)

### Tasks

#### ***Adding Subsite Flow Segments***

Perform the following steps to add one or more subsite flow segments to your job:

1. Find the folders for the subsite job in the Folders pane on the SAS Data Integration Studio desktop. Then, add an Additional Output folder and a Permanent Library folder to the folders under Data Sources folder, which is located within the Sub Site Template folder. If you are adding a segment to locate a techsupp subsite, you might call these folders Additional Output TECHSUPP and Permanent Library TECHSUPP. The name that you use here is not used in any way during the processing of the job. It simply functions as a visual cue to assist you in editing the job.
2. Add a Clickstream Parse transformation to the job flow on the **Diagram** tab of the **Job Editor** window.
3. Connect the temporary output table port of the Clickstream Parse - Global transformation to the input port of the just-added Clickstream Parse transformation.
4. Open the **General** tab in the properties window for the Clickstream Parse transformation. Then, rename the transformation to document the subsite that you need to add. For example, you can rename the transformation to Clickstream Parse - TECHSUPP if you are adding a techsupp subsite.
5. Click **Input Mapping**. Then, click **Map all columns** in the toolbar.
6. Click **Rules**. Disable the **Filter graphics files**, **Filter non-pages**, and **Filter spiders by user agent** rules, which are enabled by default. To disable the rules, click **No** in the drop-down menu in the Enable column for those rows.
7. Right-click a blank space on the **Rules** tab and click **New** in the pop-up menu. A row is added to the table.
8. Enter the following values for the columns in the new row:
  - **Enable:** Yes (via drop-down menu)

- **Group:** Subsite
- **Name:** Filter by subsite
- **When:** After Input (via drop-down menu)
- **Condition Type:** SAS expression
- **Action Type:** Delete

9. Right-click the row for the **Filter by subsite** rule. Then, click **Properties** in the pop-up menu to access the Rules Properties window.
10. Select the **SAS expression** radio button and click **Build** to access the SAS Expression Builder window. Modify a SAS expression from one of the other subsite flow segments to isolate the data from the desired subsite. For example, you can modify `(CLK_cs_URI_Stem, '/prd', 'ti')` from the PRD segment to `(CLK_cs_URI_Stem, '/techsupp', 'ti')` for the TECHSUPP segment.

*Note:* This code is an example of how you can create and modify rules to subset the data records for a subsite. You can also use multiple rules if you need them to obtain the desired result.

11. Close the row and transformation properties windows to return to the process flow.
12. Click **Control Flow** in the Details panel of the Job Editor window. Drag the subsite parse transformation for the new subsite to the position after the Checkpoint - Can we sessionize subsite data? transformation for the previous subsite in the flow. In the techsupp example, Checkpoint - Can we sessionize GEN subsite data? precedes Clickstream Parse - TECHSUPP.
13. Add a Return Code Check transformation to the job flow. Open the **General** tab in the properties window for the transformation and rename the transformation to match the parse transformation for the subsite that you just added. For example, if you are adding a techsupp subsite, you might rename the transformation to Checkpoint - Can we parse TECHSUPP subsite data?
14. Click **Status Handling**. Then, click the **Send Email** action and click **Action Options** to open the Action Options window.
15. Specify an e-mail address in the Value column for the e-mail address option. If the step fails when the subsite job is run, an e-mail message is sent to the specified address. Then, close the windows.
16. Click **Control Flow** in the Details panel of the Job Editor window. Drag the checkpoint transformation for the new subsite to the position after the Clickstream Parse - TECHSUPP transformation.
17. Add a Clickstream Sessionize transformation to the job flow. Open the **General** tab in the properties window for the transformation and rename the transformation to match the subsite that you are adding. For example, you might rename the transformation to Clickstream Sessionize -TECHSUPP.
18. Click **Options** and click **Tables** in the list at the left side of the tab.
19. Click **Browse** adjacent to the **Additional output library** field to locate the output library for the subsite. The path and name for the techsupp subsite is `/Shared Data/Sub Site Template/Data Sources/Additional Output TECHSUPP/Additional Output TECHSUPP (Library)`.
20. Click **Browse** adjacent to the **Permanent library path** field to locate the permanent library for the subsite. The path and name for the techsupp subsite is `/Shared Data/Sub Site Template/Data Sources/Permanent Library TECHSUPP/Permanent Library TECHSUPP (Library)`.



21. Click **Control Flow** in the Details panel of the Job Editor window. Drag the sessionize transformation for the new subsite to the position after the checkpoint transformation for the parse transformation. Then, drag between the temporary output table port for the parse transformation and the input port for the sessionize transformation to connect them. The sessionize transformation now has inputs from the checkpoint transformation and the parse transformation.
22. Right-click the temporary output table port for the sessionize transformation, and select the output table for the subsite flow segment from the Table Selector window.
23. Add another Return Code Check transformation to the job flow. Open the **General** tab in the properties window for the transformation and rename the transformation to match the sessionize transformation for the subsite that you just added. For example, if you are adding a techsupp subsite, rename the transformation to Checkpoint - Can we sessionize TECHSUPP subsite data?
24. Click **Status Handling**. Click the **Send Email** action, and then click **Action Options**.
25. Specify an e-mail address in the Value column for the e-mail address option. If the step fails when the subsite job is run, an e-mail message is sent to the specified address. Then, close the properties windows.
26. Click **Control Flow** in the Details panel of the Job Editor window. Drag the checkpoint transformation for the new subsite to the position after the sessionize transformation.

### ***Deleting Existing Subsite Flow Segments***

Perform the following steps to delete existing subsite flow segments:

1. Select the transformations and tables that comprise the subsite flow segment that you need to delete.
2. Right-click one of the selected objects and click **Delete** in the pop-up menu.

### ***Modifying Existing Subsite Flow Segments***

Perform the following steps to change the set of subsites isolated during the job:

1. Open the **Rules** tab in the properties window for the Clickstream Parse subsite transformation that you need to modify. Note that only the **Filter by subsite** rule is enabled.
2. Right-click the row **Filter by subsite** rule to access the Rule Properties window.
3. Modify the value in the **Expression** field under the **SAS expression** radio button to find the subsite that you need. For example, the expression `(CLK_cs_URI_Stem, '/prd', 'ti')` locates a subsite identified by PRD. Because the action specified for the rule is **Delete**, the rule isolates the subsite by filtering out all other data in the clickstream log. You can also add any additional rules that you need to control the filtering of data from the output table for this subsite.
4. Close the properties windows.

---

## **Running a Subsite Job**

### ***Problem***

You want to isolate the data from one or more subsites from a clickstream log in a job.

## Solution

You can process the clickstream log in the subsite job template. If you have not done so already, you should run a copy of the setup job for the subsite template, which is named `clk_0010_setup_sub_site`. When you process the data, you should run a copy of the subsite job, which is named `clk_0200_create_sub_site_tables`. By running a copy, you protect the original template. For information about running the setup job and creating a copy of the original job, see [“Copying the Sub Site Templates Folder” on page 50](#).

Perform the following tasks to run the template:

- [“Review and Prepare the Job” on page 54](#)
- [“Run the Job and Examine the Output” on page 55](#)

## Tasks

### Review and Prepare the Job

You can examine the subsite 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 file location of the clickstream log that you process and adjust the global rules that are applied to the log before the subsites are processed.

Perform the following steps to make these adjustments:

1. Open the renamed subsite job.
2. Scroll through the job on the **Diagram** tab.

Note the following components:

- the section that validates the source clickstream log and applies global rules
- the transformations that isolate subsites, identify sessions, and generate subsite output tables
- the section that parses the source clickstream log as whole, identifies sessions, and generates an output table for all of the log data

For an overview of how the job is processed, see [“Stages in the Subsite Template Job” on page 44](#).

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.
4. Open the **Rules** tab in the properties window for the Clickstream Parse - Global Rules transformation.

Note that the following rules are enabled:

- Filter graphics pages
- Filter non-pages
- Filter spiders by user agent

To display the properties windows for any of these rules, right-click the rule and click **Properties** in the pop-up menu. You can make any needed changes in the Rule Properties window. For example, you can edit the types of graphics files that are filtered by the **Filter graphics file rule**. Open the properties window for the rule and click

**Search Options** next to the **Column** field under the **Column search** radio button. Close the windows that you have opened before you return to the application.

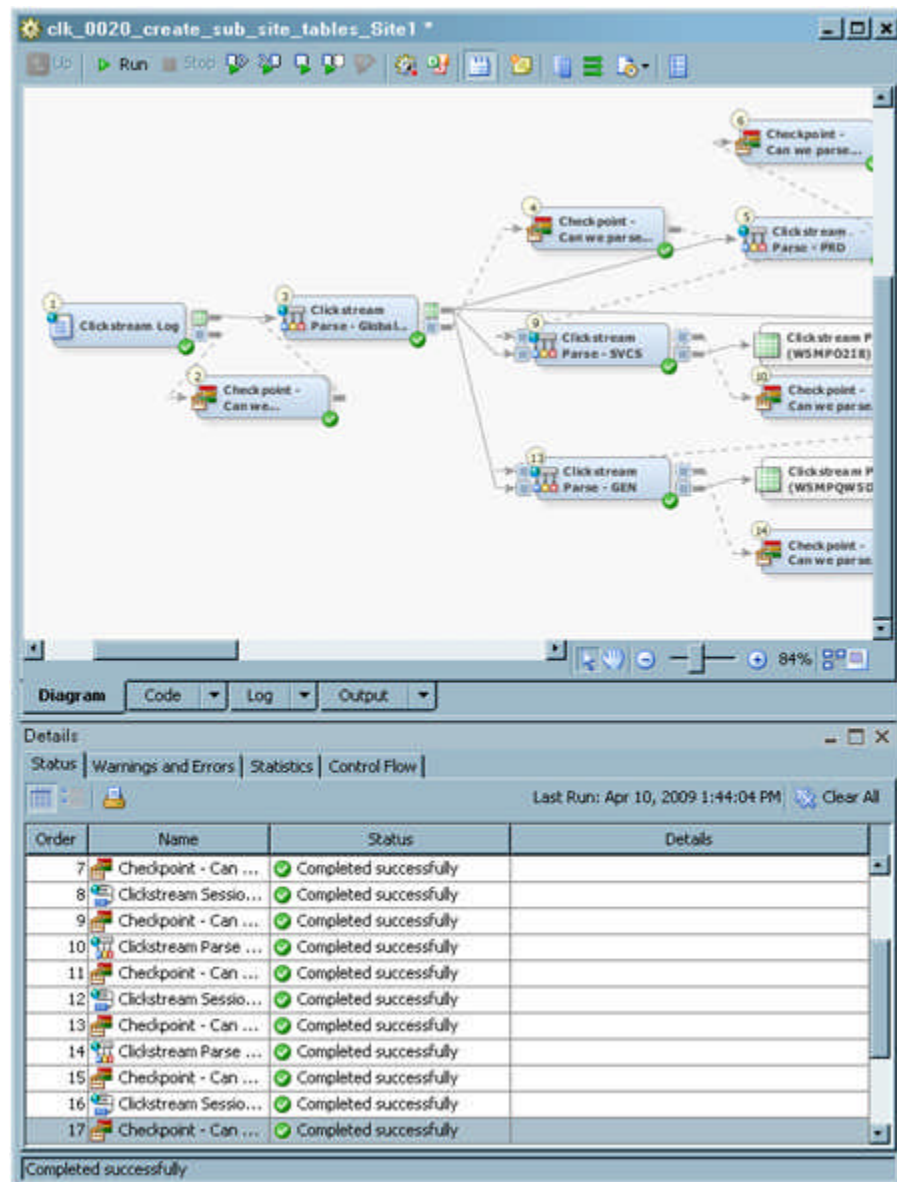
### Run the Job and Examine the Output

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

1. Run the job.

The following display shows a successfully completed sample job:

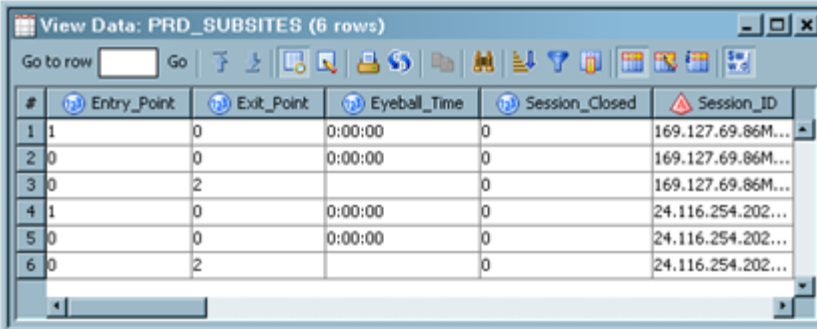
**Display 6.4** Completed Subsite Job



2. If the job completes without error, right-click the output table from one of the subsites and click **Open** in the pop-up menu.

The View Data window for the table appears, as shown in the following display:

**Display 6.5** Single Subsite Output

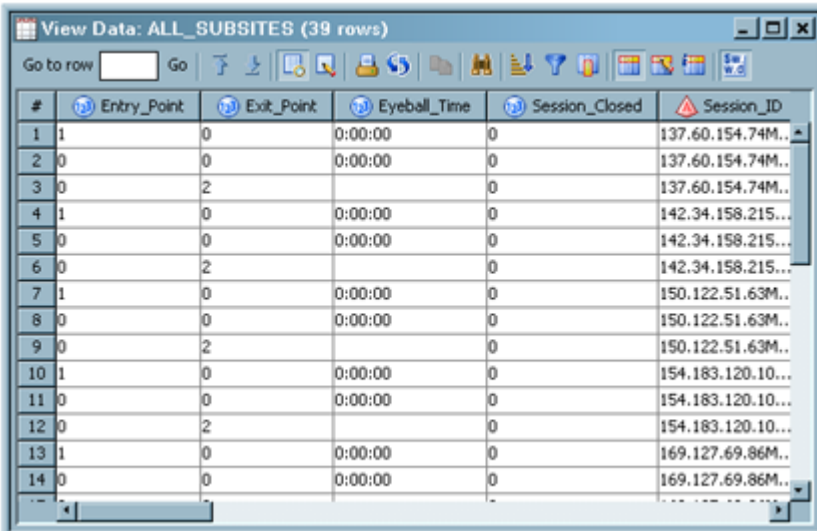


#	Entry_Point	Exit_Point	Eyeball_Time	Session_Closed	Session_ID
1	1	0	0:00:00	0	169.127.69.86M...
2	0	0	0:00:00	0	169.127.69.86M...
3	0	2		0	169.127.69.86M...
4	1	0	0:00:00	0	24.116.254.202...
5	0	0	0:00:00	0	24.116.254.202...
6	0	2		0	24.116.254.202...

- Right-click the ALL\_SUBSITES table and click **Open** in the pop-up menu.

The following display shows the View Data window for the table:

**Display 6.6** Output from All Subsites



#	Entry_Point	Exit_Point	Eyeball_Time	Session_Closed	Session_ID
1	1	0	0:00:00	0	137.60.154.74M...
2	0	0	0:00:00	0	137.60.154.74M...
3	0	2		0	137.60.154.74M...
4	1	0	0:00:00	0	142.34.158.215...
5	0	0	0:00:00	0	142.34.158.215...
6	0	2		0	142.34.158.215...
7	1	0	0:00:00	0	150.122.51.63M...
8	0	0	0:00:00	0	150.122.51.63M...
9	0	2		0	150.122.51.63M...
10	1	0	0:00:00	0	154.183.120.10...
11	0	0	0:00:00	0	154.183.120.10...
12	0	2		0	154.183.120.10...
13	1	0	0:00:00	0	169.127.69.86M...
14	0	0	0:00:00	0	169.127.69.86M...

## Chapter 7

# Processing Multiple Clickstreams

---

<b>About the Basic (Multiple) Web Log Template Job</b> . . . . .	<b>57</b>
<b>Best Practices for Multiple Log Jobs</b> . . . . .	<b>60</b>
Understanding the Propagation of Columns in the Multiple Log Template Job . . . . .	60
<b>Stages in the Basic (Multiple) Web Log Template Job</b> . . . . .	<b>60</b>
Overview . . . . .	60
Prepare Data and Parameter Values to Pass to Loop 1 . . . . .	61
Loop One: Recognize, Parse, and Group Data . . . . .	63
Combine Groups . . . . .	65
Loop Two: Sessionize . . . . .	67
Create Detail and Generate Output . . . . .	69
<b>Copying the Basic (Multiple) Web Log Templates Folder</b> . . . . .	<b>70</b>
<b>Running a Multiple Logs Job</b> . . . . .	<b>71</b>
Problem . . . . .	71
Solution . . . . .	71
Tasks . . . . .	71

---

## About the Basic (Multiple) Web Log Template Job

The Basic (Multiple) Web Log Template provides you with a parameterized version of a simple template job that enables you process multiple clickstream logs from the same or multiple servers. It also enables you to optimize processing time through the use of symmetric multi-processing using SAS MP Connect or grid computing. Finally, the template manages outputs and resources to avoid contention.

The Multiple Log Template job uses the same Clickstream Log, Clickstream Parse, and Clickstream Sessionize transformations as are used in the single log template job. 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.

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 upon 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 7.1 on page 59](#).

*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 7.1 on page 59](#). 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.

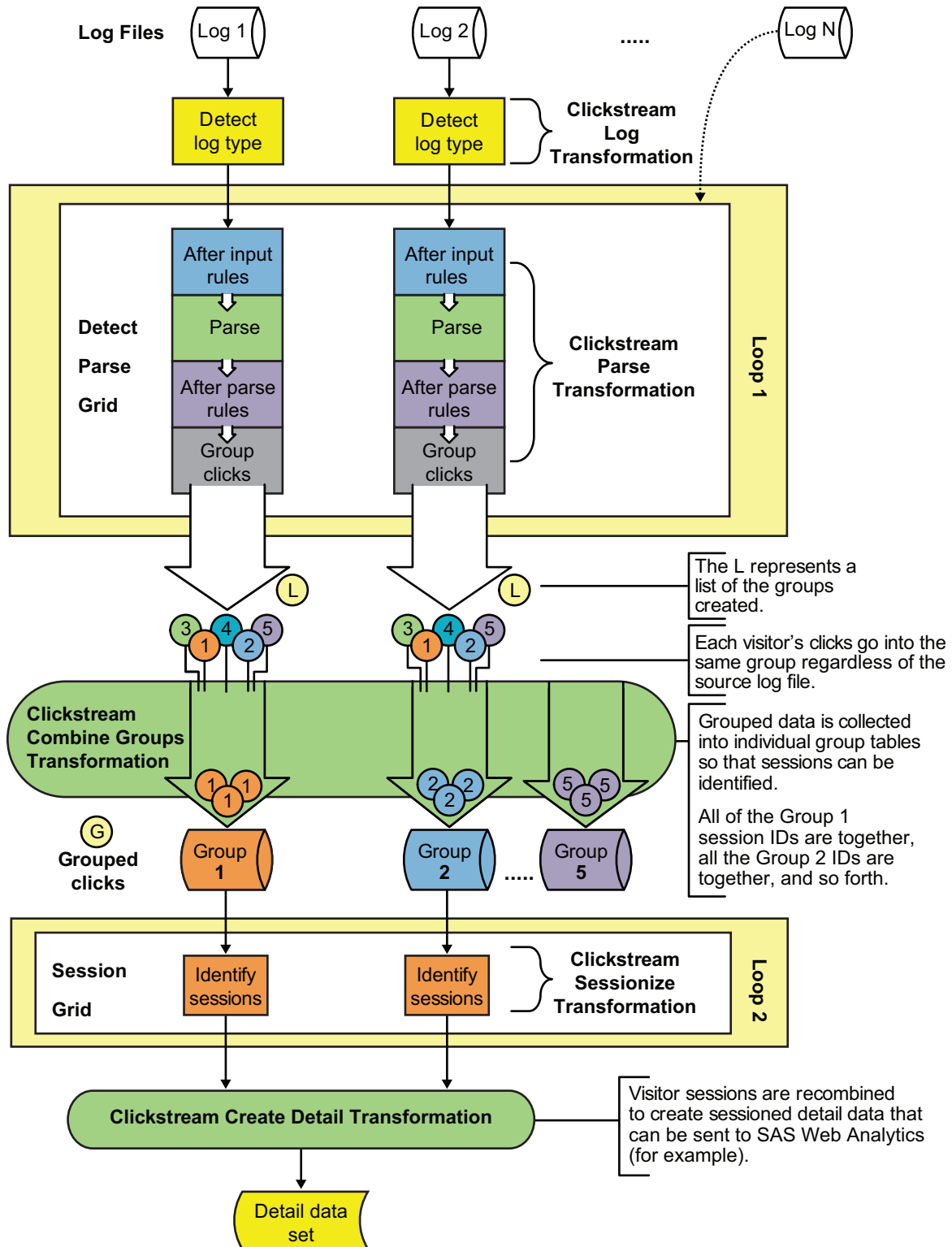
The completion of the visitor ID ensures that the visitor ID value that is assigned to users after they log on is present on every record of the session. This persistence holds even when the users browse the site for a period of time before logging in and after they log out. The visitor ID value is useful 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 and 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 multiple clickstream log jobs.

**Figure 7.1** Multiple Log Job Process Flow



Sections of this figure are included in the descriptions of each stage of the template's processing.

---

## Best Practices for Multiple Log Jobs

### *Understanding the Propagation of Columns in the Multiple Log Template Job*

SAS Data Integration Studio can automatically propagate columns from one transformation to another. However, this propagation is not necessarily possible in the Basic (Multiple) Web Log Template Job because of the use of 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 MULTI\_DDS\_OUTPUT. 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 MULTI\_DDS\_OUTPUT 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 MULTI\_DDS\_OUTPUT table.

---

## Stages in the Basic (Multiple) Web Log Template Job

### *Overview*

The Basic (Multiple) Web Log Template Job can be divided into the following stages:

- [“Prepare Data and Parameter Values to Pass to Loop 1” on page 61](#)
- [“Loop One: Recognize, Parse, and Group Data” on page 63](#)
- [“Combine Groups” on page 65](#)
- [“Loop Two: Sessionize” on page 67](#)



- “Create Detail and Generate Output” on page 69

## 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.

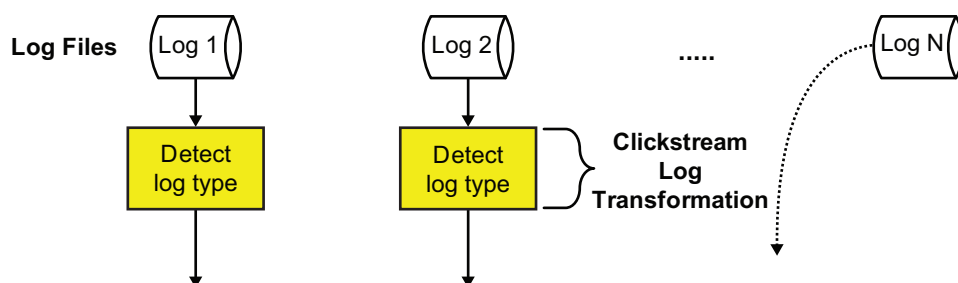
### NUMGROUPS

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 multiple log template process locates the data and parses it.

The following figure illustrates this stage of the process:

**Figure 7.2** Locate and Parse Data



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

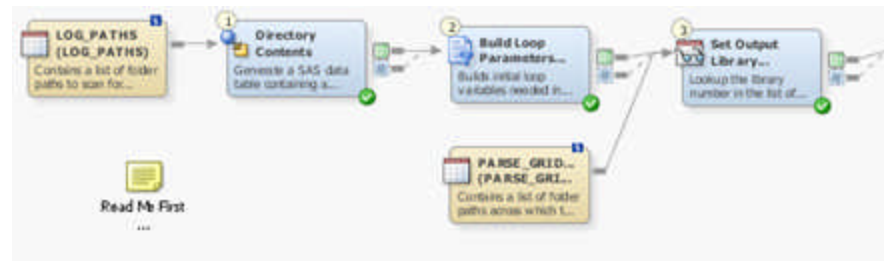
**Table 7.1** Locate and Parse Transformations and Tables

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

Name	Description	Inputs from and Outputs to
Directory Contents transformation	<p>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:</p> <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>	<p>From: LOG_PATHS table</p> <p>To: Build Loop Parameters (reused SAS Extract) transformation</p>
Build Loop Parameters (reused SAS Extract) transformation	<p>Passes through the columns from the Directory Contents transformation and creates two additional columns. LIBRARYNUMBER is a number from 1 to <math>n</math> where <math>n</math> 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.</p>	<p>From: Directory Contents transformation</p> <p>To: Set Output Library Locations (reused Lookup) transformation</p>
PARSE_GRID_PATHS table	<p>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.</p>	<p>From: None</p> <p>To: Set Output Library (reused SAS Extract) transformation</p>
Set Output Library (reused SAS Extract) transformation	<p>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.</p>	<p>From: Build Loop Parameters (reused SAS Extract) transformation, and PARSE_GRID_PATHS table</p> <p>To: Loop 1 (Recognize and Parse) transformation</p>

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

**Display 7.1** Locate and Parse Process Flow

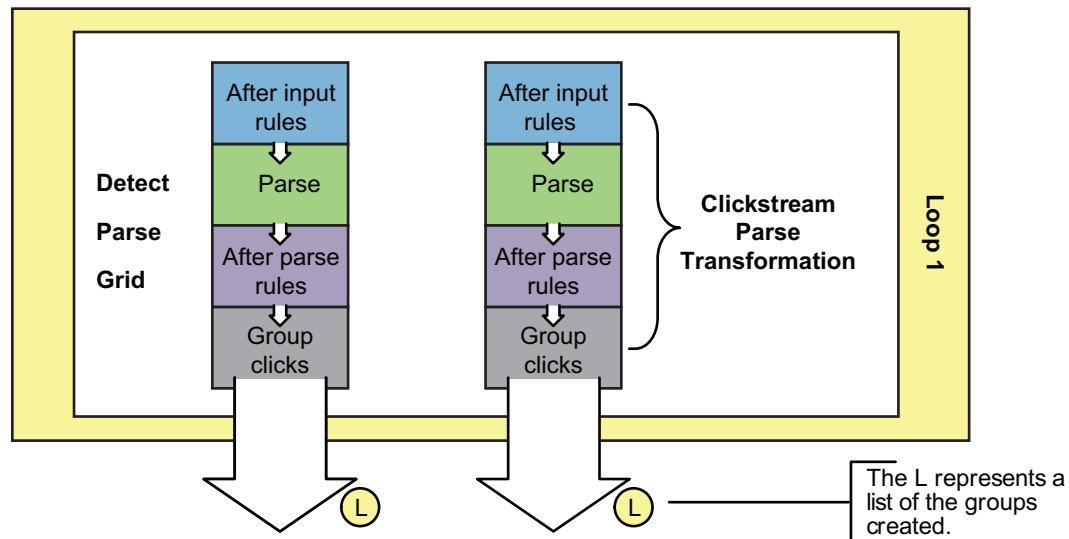


### Loop One: Recognize, Parse, and Group Data

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. 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 enable you to configure how errors are processed.

The following figure illustrates this stage of the process:

**Figure 7.3** Loop One: Recognize, Parse, and Group Data



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

**Table 7.2** 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 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: Checkpoint - Can we recognize the log? transformation</p> <p>To: Clickstream Parse transformation</p>
Checkpoint - Can we recognize the 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>From: Checkpoint - Can we recognize the log? transformation</p> <p>To: Checkpoint - Parse OK? transformation</p>
Checkpoint - Parse OK? 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: Checkpoint - Parse OK? transformation  To: Filter - Only properly parsed logs (reused SAS Extract) transformation

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

**Display 7.2** Loop 1 Process Flow

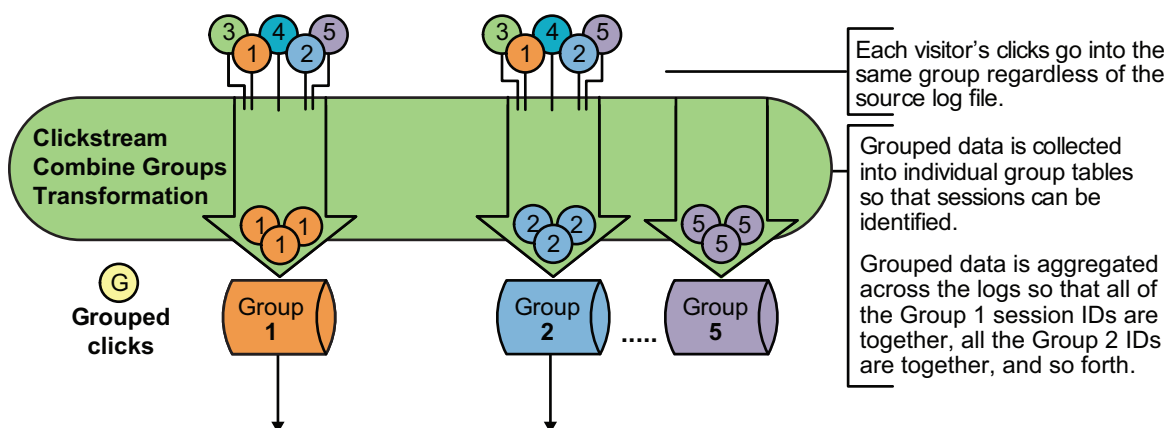


## Combine Groups

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 7.4** Combine Groups



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

**Table 7.3** 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 7.3** Combine Groups Process Flow

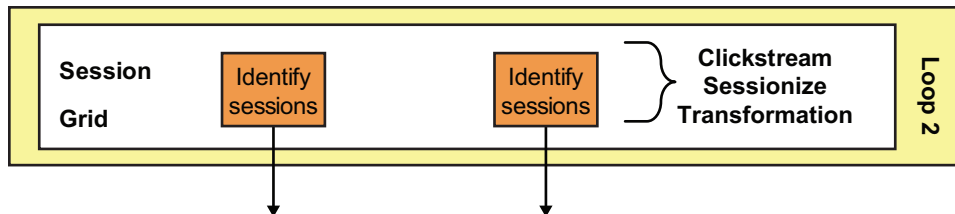


## Loop Two: Sessionize

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 7.5** Loop Two: Sessionize



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

**Table 7.4** 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>

Name	Description	Inputs from and Outputs to
PARAM_PARSE_RESULTS table	A parameterized table for receiving the output from the Clickstream Parse transformation and passing it into the Clickstream Sessionize transformation. (See “ <a href="#">Understanding the Propagation of Columns in the Multiple Log Template Job</a> ” on page 60 if you have defined User Columns that need to be propagated to the final detail table.)	From: None To: Clickstream Sessionize transformation
Clickstream Sessionize transformation	Identifies sessions in the grouped data.	From: Loop 2 (Identify Sessions) transformation and PARAM_PARSE_RESULTS table To: Checkpoint - Can we identify sessions? transformation and CLICKSTREAM_SESSIONIZE table
CLICKSTREAM_SESSIONIZE table	Stores CLICKSTREAM_SESSIONIZE output and ensures the sort sequence of the output data is correct. (See “ <a href="#">Backing Up PERMLIB</a> ” on page 30.)	From: Clickstream Sessionize transformation To: None
Checkpoint - Can we identify sessions? 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: Checkpoint - Can we identify sessions? transformation To: Filter Failed Jobs (SAS Extract) transformation

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

**Display 7.4** Loop 2 Process Flow



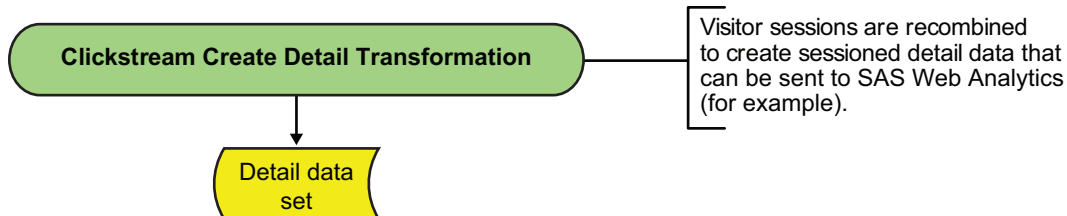


## Create Detail and Generate Output

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 7.6** Create Detail and Generate Output



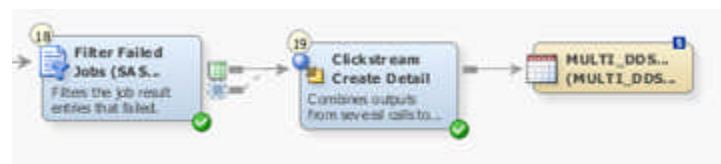
The transformations and tables in this stage are described in the following table:

**Table 7.5** Detail and Output Transformations

Name	Description	Inputs from and Outputs to
Filter Failed Jobs (SAS Extract) transformation	Uses the status table generated by the Loop transformation to determine which sub-jobs 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 Failed Jobs (SAS Extract) transformation To: MULTI_DDS_OUTPUT table
MULTI_DDS_OUTPUT 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 7.5** Create Detail and Generate Output Process Flow



## Copying the Basic (Multiple) Web Log Templates Folder

You should copy the Basic (Multiple) Web Log Templates folder before you modify any of the objects 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. Perform the following steps to copy and prepare the Basic (Multiple) Web Log Templates folder:

1. Right-click the **Basic Web Log Template** folder in the **Multiple Log Templates** folder. Then, select **Copy** from the pop-up menu.
2. Right-click the folder where you want to paste the template. Then, select **Paste Special** from the pop-up menu 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:* 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 for objects such as SAS libraries. Click **Finish** when you complete the pages.

3. Rename (if desired) and expand the new **Basic Web Log Template** folder that was just copied. Then, open the properties window for the two jobs in the 2.1 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\_multi job to clk\_0010\_setup\_basic\_multi\_Site1 and the clk\_0020\_load\_multi\_dds job to clk\_0020\_load\_multi\_dds\_Site1.
4. Expand the Data Sources for the template and its subfolders to reveal the libraries used by the multiple logs job. To distinguish these libraries from the original libraries used by the Basic (Multiple) Web Log Templates job, you can rename these libraries to include the site name. For example, you can rename the Multi - Additional Output folder to Multi - Additional Output - Site1.
5. If you modified the directory paths when copying the basic Web log templates you need to open the renamed clk\_0010\_setup\_basic\_multi 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.
6. Run the renamed clk\_0010\_setup\_basic\_multi job. This job creates the necessary folder structure on the file system and generates sample data to support the renamed clk\_0020\_load\_multi\_dds job.
7. Open the job properties window in the renamed clk\_0020\_load\_multi\_dds job. Then, edit the EMAILADDRESS parameter on 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.

---

## Running a Multiple Logs Job

### **Problem**

You want to process the data contained in more than one clickstream log in a single job. You also want to improve performance by using parallel processing.

### **Solution**

You can process the job in the multiple log job template. If you have not done so already, you should run a copy of the setup job for the multiple logs template, which is named `clk_0010_setup_basic_multi_job`. When you actually process the data, you should run a copy of the multiple logs job, which is named `clk_0200_load_multi_dds`. By running a copy, you protect the original template. For information about running the setup job and creating a copy of the original job, see [“Copying the Basic \(Multiple\) Web Log Templates Folder” on page 70](#).

Perform the following tasks to run the template:

- [“Review and Prepare the Job” on page 71](#)
- [“Run the Job and Examine the Output” on page 72](#)

### **Tasks**

#### ***Review and Prepare the Job***

You can examine the multiple logs 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, set the number of groups that are used in the sessionizing loop, and specify parallel and multiple processing options.

Perform the following steps to make these adjustments:

1. Open the renamed multiple logs 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 the Basic \(Multiple\) Web Log Template Job” on page 60](#).

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 the two parameters **Number of Distinct Clickstream Parse Output Paths** and **Number of Groups into which data should be divided** for the job. 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.

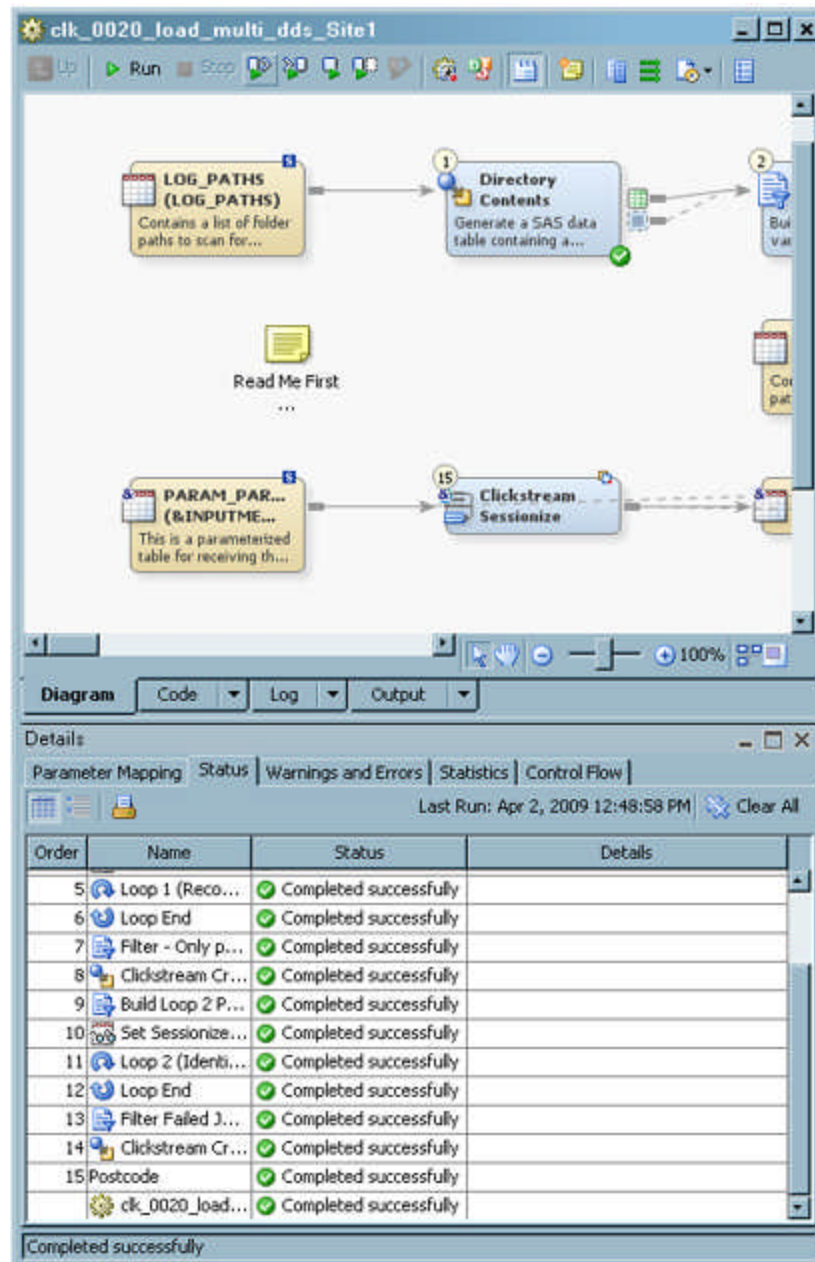
### **Run the Job and Examine the Output**

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

1. Run the job.

The following display shows a successfully completed sample job.

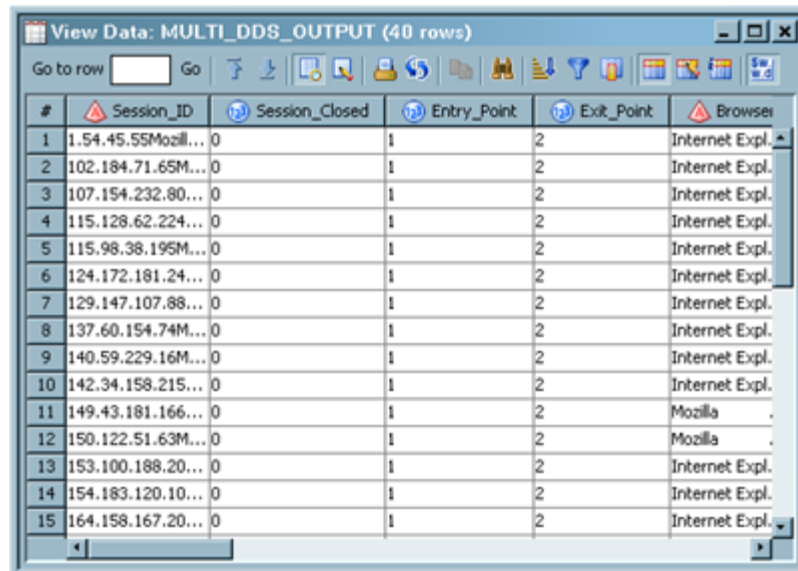
**Display 7.6** Completed Multiple Log Job



- If the job completes without error, right-click the MULTI\_DDS\_OUTPUT 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 7.7** Multiple Log Job Output



#	Session_ID	Session_Closed	Entry_Point	Exit_Point	Browser
1	1.54.45.55Mozill...	0	1	2	Internet Expl.
2	102.184.71.65M...	0	1	2	Internet Expl.
3	107.154.232.80...	0	1	2	Internet Expl.
4	115.128.62.224...	0	1	2	Internet Expl.
5	115.98.38.195M...	0	1	2	Internet Expl.
6	124.172.181.24...	0	1	2	Internet Expl.
7	129.147.107.88...	0	1	2	Internet Expl.
8	137.60.154.74M...	0	1	2	Internet Expl.
9	140.59.229.16M...	0	1	2	Internet Expl.
10	142.34.158.215...	0	1	2	Internet Expl.
11	149.43.181.166...	0	1	2	Mozilla
12	150.122.51.63M...	0	1	2	Mozilla
13	153.100.188.20...	0	1	2	Internet Expl.
14	154.183.120.10...	0	1	2	Internet Expl.
15	164.158.167.20...	0	1	2	Internet Expl.

If the job does not complete successfully, then you might want to examine the logs for each loop in the job. Since most of the processing is done in the loop portion of the job, this is where most errors occur. 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).

## Chapter 8

# Processing Campaign Information

---

<b>About the Customer Integration Template Job</b> . . . . .	<b>75</b>
<b>Stages in the Customer Integration Template Job</b> . . . . .	<b>76</b>
Overview . . . . .	76
Prepare Data and Parameter Values to Pass to Loop 1 . . . . .	76
Loop One: Recognize, Parse, and Group Data . . . . .	78
Combine Groups . . . . .	79
Loop Two: Sessionize . . . . .	81
Create Detail and Generate Output . . . . .	82
<b>Copying the Customer Integration Template Folder</b> . . . . .	<b>83</b>
<b>Collecting Campaign Information in a Customer Integration Job</b> . . . . .	<b>84</b>
Problem . . . . .	84
Solution . . . . .	85
Tasks . . . . .	85

---

## About the Customer Integration Template Job

You can use the Customer Integration Template job to capture information that enables customer Web-based activity to be associated with the marketing campaign that originated the activity. Once campaign information is passed to the SAS Digital Marketing redirection servlet (SDM's tracking servlet), the Response Tracking Code (RTC) and Subject ID (Sn) values are not currently forwarded as part of any subsequent requests.

The Customer Integration 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.

The Customer Integration Template job is similar to the Basic (Multiple) Web Log Template job. For detailed information about how the Basic (Multiple) Web Log Template job works, see [“About the Basic \(Multiple\) Web Log Template Job” on page 57](#).

*Note:* This feature is available only in the maintenance release of SAS Data Surveyor for Clickstream Data 2.1.

## Stages in the Customer Integration Template Job

### Overview

The Customer Integration Template job can be divided into the following stages:

- “Prepare Data and Parameter Values to Pass to Loop 1” on page 76
- “Loop One: Recognize, Parse, and Group Data” on page 78
- “Combine Groups” on page 79
- “Loop Two: Sessionize” on page 81
- “Create Detail and Generate Output” on page 82

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

The Read Me First note in the job flow contains information that is recommended for the initial setup and modification of this job. You might also 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 transformation 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 campaign information template process locates the data and parses it.

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

**Table 8.1** Locate and Parse Transformations and Tables

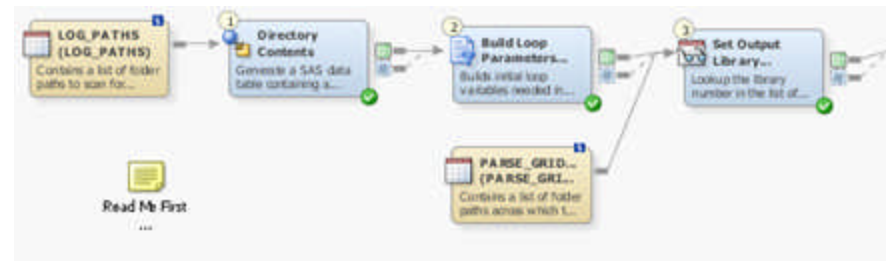
Name	Description	Inputs from and Outputs to
LOG_PATHS table	Contains a list of folder paths to scan for clickstream logs.	From: None To: Directory Contents transformation



Name	Description	Inputs from and Outputs to
Directory Contents transformation	<p>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:</p> <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>	<p>From: LOG_PATHS table</p> <p>To: Build Loop Parameters (reused SAS Extract) transformation</p>
Build Loop Parameters (reused SAS Extract) transformation	<p>Passes through the columns from the Directory Contents transformation and creates two additional columns. LIBRARYNUMBER is a number from 1 to <math>n</math> where <math>n</math> 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.</p>	<p>From: Directory Contents transformation</p> <p>To: Set Output Library Locations (reused Lookup) transformation</p>
PARSE_GRID_PATHS table	<p>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.</p>	<p>From: None</p> <p>To: Set Output Library (reused SAS Extract) transformation</p>
Set Output Library (reused SAS Extract) transformation	<p>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.</p>	<p>From: Build Loop Parameters (reused SAS Extract) transformation, and PARSE_GRID_PATHS table</p> <p>To: Loop 1 (Recognize and Parse) transformation</p>

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

**Display 8.1** Locate and Parse Process Flow



### Loop One: Recognize, 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. Each stream consists of a Clickstream Log transformation, a Clickstream Parse transformation, and two checkpoints that are created by renaming the Return Code transformation and that enable you to configure how errors are processed.

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

**Table 8.2** 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 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: Checkpoint - Can we recognize the log? transformation</p> <p>To: Clickstream Parse transformation</p>
Checkpoint - Can we recognize the 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>

Name	Description	Inputs from and Outputs to
Clickstream Parse transformation	<p>Parses this data, identifies the campaign and customer who clicked on a specific treatment, 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>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: Checkpoint - Can we recognize the log? transformation</p> <p>To: Checkpoint - Parse OK? transformation</p>
Checkpoint - Parse OK? transformation	Evaluates the return code from Clickstream Parse; sends e-mail to specified address if the parse step fails.	<p>From: Clickstream Parse transformation</p> <p>To: Loop End transformation</p>
Loop End transformation	Ends loop processing; returns to beginning of loop	<p>From: Checkpoint - Parse OK? transformation</p> <p>To: Filter - Only properly parsed logs (reused SAS Extract) transformation</p>

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

**Display 8.2** Loop 1 Process Flow



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

**Table 8.3** 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 therefore should 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 8.3** 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 transformations and tables in this stage are described in the following table:

**Table 8.4** 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="#">“Understanding the Propagation of Columns in the Multiple Log Template Job”</a> on page 60 if you have defined User Columns that need to be propagated to the final detail table.)</p> <p>This table contains the columns that support campaign information.</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: Checkpoint - Can we identify sessions? 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 30.)	From: Clickstream Sessionize transformation To: None
Checkpoint - Can we identify sessions? 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: Checkpoint - Can we identify sessions? transformation To: Filter Failed Jobs (SAS Extract) transformation

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

**Display 8.4** 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 transformations and tables in this stage are described in the following table:

**Table 8.5** Detail and Output Transformations

Name	Description	Inputs from and Outputs to
Filter Failed Jobs (SAS Extract) transformation	Uses the status table generated by the Loop transformation to determine which subjobs were successful and therefore should 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 Failed Jobs (SAS Extract) transformation To: CI_DDS_OUTPUT table
CI_DDS_OUTPUT table	Contains the output from the 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 8.5** Create Detail and Generate Output Process Flow



## Copying the Customer Integration Template Folder

You should copy the Customer Integration Template folder before you modify any of the objects 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. Perform the following steps to copy and prepare the multiple log template:

1. Right-click the **Customer Integration Template** folder that is located in the **Basic (Multiple) Web Log Templates** folder. Then, select **Copy** from the pop-up menu.
2. Right-click the folder where you want to paste the template. Then, select **Paste Special** from the pop-up menu 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:* 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 for objects such as SAS libraries. Click **Finish** when you complete the pages.

3. Rename (if desired) and expand the new **Customer Integration Template** folder that was just copied. Then, open the properties window for the two jobs in the 2.1 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_ci_job` to `clk_0010_setup_basic_ci_job_site1` and the `clk_0020_load_ci_dds_job` to `clk_0020_load_ci_dds_site1`.
4. Expand the Data Sources for the template and its subfolders to reveal the libraries used by the Customer Integration job. To distinguish these libraries from the original libraries used by the Customer Integration Template job, you can rename these libraries to include the site name. For example, you can rename the CI - Additional Output folder to CI - Additional Output - Site1.
5. If you modified the directory paths when copying the basic Web log templates, then you need to open the renamed `clk_0010_setup_basic_ci_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.
6. Run the renamed `clk_0010_setup_basic_ci_job` job. This job creates the necessary folder structure on the file system and generates sample data to support the renamed `clk_0020_load_ci_dds_job`.
7. Open the job properties window in the renamed `clk_0020_load_ci_dds_job`. Then, edit the EMAILADDRESS parameter on the **Parameters** tab as follows:
  - a. Select the EMAILADDRESS row in the table.
  - b. Click **Edit** to access the Edit Prompt window.
  - c. Click **Prompt Type and Value** and enter the e-mail address to use for any failure notification messages in the **Default value** field.
  - d. Click **OK** to exit the job properties window.

---

## Collecting Campaign Information in a Customer Integration Job

### **Problem**

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.



## Solution

You can collect campaign information by using the Customer Integration Template job. This template is virtually identical to the Basic (Multiple) Web Log Template. The only difference is that new columns have been added to contain campaign information, Clickstream Parse rules are added to extract the values from the raw Web log, and the Fill Column options are used to copy the values for these new columns into all records for a session.

If you have not done so already, you should run a copy of the setup job for the Customer Integration Template job, which is named `clk_0010_setup_basic_ci_job`. When you actually process the data, you should run a copy of the Customer Integration Template job, which is named `clk_0200_load_ci_dd`s. By running a copy, you protect the original template. For information about running the setup job and creating a copy of the original job, see [“Copying the Customer Integration Template Folder” on page 83](#).

Perform the following tasks to run the template:

- [“Review and Prepare the Job” on page 85](#)
- [“Set Campaign Information Options” on page 86](#)
- [“Run the Job and Examine the Output” on page 86](#)

## Tasks

### **Review and Prepare the Job**

You can examine the Customer Integration 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, set the number of groups that are used in the sessionizing loop, and specify parallel and multiple processing options.

Perform the following steps to make these adjustments:

1. Open the renamed multiple logs 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 information about how the job is processed, see [“About the Customer Integration Template Job” on page 75](#).

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** icon 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 the two parameters **Number of Distinct Clickstream Parse Output Paths** and **Number of Groups into which data should be divided** for the job. 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** as necessary to close the dialog boxes and 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.

### **Set Campaign Information Options**

Perform the following steps to set options that enable you to capture campaign information:

1. Open the properties window of the Clickstream Sessionize transformation.
2. Review the **Forward fill columns** and **Complete fill columns** options to verify that they are set appropriately for your needs.
3. Click **OK** to save the option settings and close the properties window.

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

Perform the following steps to run a Customer Integration Template job and examine its output:

1. Run the job.

The following display shows a successfully completed sample job.

**Display 8.6** Completed Customer Integration Template Job

The screenshot displays the SAS Enterprise Guide interface for a job titled 'clk\_0020\_load\_ci\_dd\_site1'. The job diagram shows a sequence of steps: 'LOG\_PATHS (LOG\_PATHS)' (Step 1), 'Directory Contents' (Step 2), 'PARAM\_PAR... (&INPUTME...' (Step 15), and 'Clickstream Sessionize' (Step 15). The 'Details' pane at the bottom provides a comprehensive log of the job execution, showing 15 steps, all of which completed successfully.

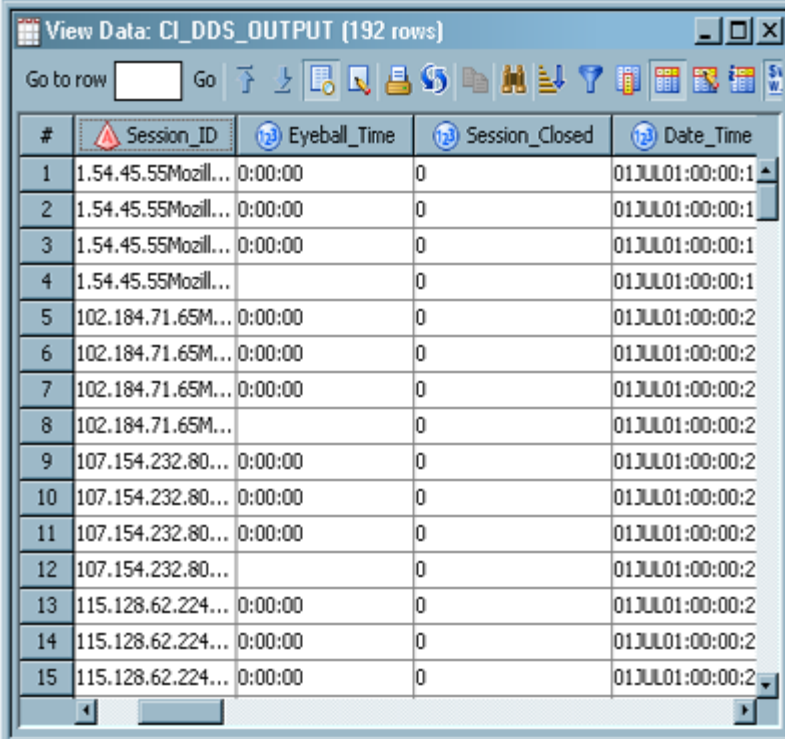
Order	Name	Status	Details
1	Precode	Completed successfully	
2	Directory Con...	Completed successfully	
3	Build Loop Par...	Completed successfully	
4	Set Output Li...	Completed successfully	
5	Loop 1 (Reco...	Completed successfully	
6	Loop End	Completed successfully	
7	Filter - Only p...	Completed successfully	
8	Clickstream C...	Completed successfully	
9	Build Loop 2 P...	Completed successfully	
10	Set Sessioniz...	Completed successfully	
11	Loop 2 (Identi...	Completed successfully	
12	Loop End	Completed successfully	
13	Filter Failed J...	Completed successfully	
14	Clickstream C...	Completed successfully	
15	Postcode	Completed successfully	
	clk_0020_loa...	Completed successfully	

Completed successfully

- If the job completes without error, right-click the CI\_DDS\_OUTPUT 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 8.7** Customer Integration Template Job Output



#	Session_ID	Eyeball_Time	Session_Closed	Date_Time
1	1.54.45.55Mozill...	0:00:00	0	01JUL01:00:00:1
2	1.54.45.55Mozill...	0:00:00	0	01JUL01:00:00:1
3	1.54.45.55Mozill...	0:00:00	0	01JUL01:00:00:1
4	1.54.45.55Mozill...	0:00:00	0	01JUL01:00:00:1
5	102.184.71.65M...	0:00:00	0	01JUL01:00:00:2
6	102.184.71.65M...	0:00:00	0	01JUL01:00:00:2
7	102.184.71.65M...	0:00:00	0	01JUL01:00:00:2
8	102.184.71.65M...	0:00:00	0	01JUL01:00:00:2
9	107.154.232.80...	0:00:00	0	01JUL01:00:00:2
10	107.154.232.80...	0:00:00	0	01JUL01:00:00:2
11	107.154.232.80...	0:00:00	0	01JUL01:00:00:2
12	107.154.232.80...	0:00:00	0	01JUL01:00:00:2
13	115.128.62.224...	0:00:00	0	01JUL01:00:00:2
14	115.128.62.224...	0:00:00	0	01JUL01:00:00:2
15	115.128.62.224...	0:00:00	0	01JUL01:00:00:2

The campaign-specific fields are found at the end of the field list as shown in the following display.

Domain	Query_String	EntryActionID	EntrySource	S1	S2	
...	...			...	...	
...	...	12345678901234	SMD	123	...	abc
...	rtc=12345678901...	12345678901234	SMD	123	...	abc
...	...	12345678901234	SMD	123	...	abc
...	s1=456	12345678901234	SMD	456	...	abc
...	...	12345678901234	SMD	456	...	abc
...	s1=789&s2=xyz	12345678901234	SMD	789	...	xyz
...	...	12345678901234	SMD	789	...	xyz
...	...	12345678901234	SMD	789	...	xyz
...	s1=333&s2=bbb	12345678901234	SMD	333	...	bbb
...	...	12345678901234	SMD	333	...	bbb
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...			...	...	
...	...	43210987654321	SMD	321	...	abc
...	rtc=43210987654...	43210987654321	SMD	321	...	abc
...	s1=987&s2=abc	43210987654321	SMD	987	...	abc

If the job does not complete successfully, then you might want to examine the logs for each loop in the job. Since most of the processing is done in the loop portion of the job, this is where most errors occur. 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 Customer Integration 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).



## Chapter 9

# Processing Tagged Pages

---

<b>About Tagging Web Pages</b> .....	<b>92</b>
<b>Best Practices for Page Tagging Jobs</b> .....	<b>92</b>
Evaluating Security Issues for Form Capture .....	92
Securing Clickstream Collection Server Log Files .....	93
<b>Preparing the Clickstream Collection Server</b> .....	<b>93</b>
<b>Copying the Page Tagging Template</b> .....	<b>93</b>
<b>JavaScript Page Tag Code</b> .....	<b>94</b>
<b>Inserting a Minimal Tag</b> .....	<b>95</b>
Problem .....	95
Solution .....	95
Tasks .....	95
<b>Inserting a Full Page Tag</b> .....	<b>96</b>
Problem .....	96
Solution .....	96
Tasks .....	96
<b>Customizing a Full Page Tag</b> .....	<b>98</b>
Overview .....	98
Debug Mode .....	98
Page Load Tracking .....	98
Predefined Data Elements .....	98
User-Defined Data Elements .....	100
Meta Tag Tracking .....	101
Cookie Tracking .....	101
Link Tracking .....	102
Form Tracking .....	103
Rich Internet Application Tracking .....	104
<b>Configuring Link Tracking in Tagged Pages</b> .....	<b>104</b>
Problem .....	104
Solution .....	104
Tasks .....	105
<b>Running a Page-Tagging ETL Job</b> .....	<b>107</b>
Problem .....	107
Solution .....	107
Tasks .....	108

---

## About Tagging Web Pages

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*.

The page tag runs inside of the user's Web browser when the user accesses a tagged page. The 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 standard 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 "JavaScript Page Tag Code" on page 94.

Working together, the 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 page tagging code that you insert. For more information, see the *SAS Data Surveyor for Clickstream Data 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream21M1/en/>.

---

## Best Practices for Page Tagging Jobs

### *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 upon how the page tag is configured for a site, sensitive information can be collected about user activity. All data collected by the 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 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 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.

---

## Copying the Page Tagging Template

You should copy the Page Tagging Template folder 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. Perform the following steps to copy and prepare the page tagging template:

1. Right-click the **Page Tagging Template** folder. Then, click **Copy** in the pop-up menu.
2. Right-click the folder where you want to paste the template. Then, click **Paste Special** in the pop-up menu 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 copy.

*Note:* The decision to select **Paste Special** rather than **Paste** is very important. If you select **Paste**, then the paths in your copied job 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 for objects such as SAS libraries. Click **Finish** when you complete the pages.

3. Rename (if desired) and expand the new Page Tagging Template folder that was just copied. Then, open the properties window and rename the two jobs in the 2.1 Jobs folder. 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\_page\_tagging job to clk\_0010\_setup\_page\_tagging\_Site1 and the clk\_0020\_page\_tagging\_detail job to clk\_0020\_page\_tagging\_detail\_Site1.
4. Expand the Data Sources folder and its subfolders to reveal the libraries used by the page tagging job. To distinguish these libraries from the original libraries used by the Page Tagging Template job, you can rename these libraries to include the site name. For example, you can rename the Additional Output - Tag library to Site1 - Additional Output - Tag.
5. If you modified the directory paths when copying the multiple log templates, then open the renamed clk\_0010\_setup\_page\_tagging job and modify the Setup transformation properties. (Otherwise, proceed to step 6.) Then, in the **Options** tab, modify the values in the **Root Directory** and **Template DirectoryName** 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.
6. Run the renamed clk\_0010\_setup\_page\_tagging job. This job creates the necessary folders and sample data to support the renamed clk\_0020\_page\_tagging\_detail job.
7. Open the job properties window in the renamed clk\_0020\_page\_tagging\_detail job. Then, edit the EMAILADDRESS parameter on the **Parameters** tab.
  - First, select the EMAILADDRESS row in the table.
  - Second, click **Edit** to access the Edit Prompt window.
  - Third, click on the **Prompt Type and Value** tab 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.
8. Open the properties window for the Clickstream Log transformation and specify the appropriate value in the **File name** field on the **File Location** tab.

At this point, you are ready to run the job.

---

## JavaScript Page Tag Code

Once you have put one or more operational clickstream collection servers in place, the page tag code needs to be inserted into the pages of interest for data collection. The data that is collected by a page tag is said to be tracked. The SAS Data Surveyor for Clickstream Data provides a robust page tag that allows 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 98](#).

Most of the time, you insert a full 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 full page tags, see [“Inserting a Full Page Tag” on page 96](#). For information about customized data tracking, see [“Customizing a Full Page Tag” on page 98](#). You can also insert a minimal page tag that includes only a set of required lines to enable default data tracking. For information about minimal page tags, see [“Inserting a Minimal Tag” on page 95](#).

---

## Inserting a Minimal Tag

### Problem

You want to insert the minimal amount of code on each Web page, and you are interested in the default data that is collected.

### Solution

You can insert a minimal 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 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, .aspx, .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 Page Tag” on page 96](#) in the *SAS Data Surveyor for Clickstream Data 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

### Tasks

#### **Insert a Minimal 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), the `https` prefix should be used instead of `http`.

---

## Inserting a Full Page Tag

### Problem

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. You also want to use debug mode during initial integration and testing of the tag code that you are inserting into your Web pages.

### Solution

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. Most tagging implementations use full page tags.

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

### Tasks

#### **Insert a Full 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), the https prefix should be used instead of http.

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

**Table 9.1** Line-By-Line Explanation of the Full 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 page tag, see [“Customizing a Full Page Tag” on page 98](#).

---

## Customizing a Full 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 98](#)
- [“Page Load Tracking” on page 98](#)
- [“Predefined Data Elements” on page 98](#)
- [“User-Defined Data Elements” on page 100](#)
- [“Meta Tag Tracking” on page 101](#)
- [“Cookie Tracking” on page 101](#)
- [“Link Tracking” on page 102](#)
- [“Form Tracking” on page 103](#)
- [“Rich Internet Application Tracking” on page 104](#)

### 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 `SAS Tag Debug.js` into the page tag code. For more information about this line, see [“Inserting a Full Page Tag” on page 96](#).

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 that is being 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.

### 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. However, exceptions are documented in the following table, which lists the predefined data elements,

including the key name, value, and default enabled status. The following table lists the predefined elements that can be collected:

**Table 9.2** Predefined Data Elements

Key Name	Description	Value	Default Enabled Status
VER	Displays the version number indicating the way the tag data is written.	2.1 (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
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

Key Name	Description	Value	Default Enabled Status
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 selecting the **Submit** button.

### User-Defined Data Elements

In cases where the predefined data elements do not provide enough information, user-defined data elements can be tracked. Code to track 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. The benefit of using an externally included configuration file such as SASSiteConfig.js is that the tagging code on each page does not have to be edited 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>. 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

## 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

## Link Tracking

A link on a page is tracked if the 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 104 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

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 optionally 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 2.1 Page Tagging JavaScript Reference* at .

## Form Tracking

A form on a page is tracked if the 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/en/>.

*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, 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. Additionally, 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 92.

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.

## 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 2.1 Page Tagging JavaScript Reference* at <http://support.sas.com/rnd/gendoc/clickstream/21M1/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 when the user clicks the left mouse button 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 movement passes an EVNT parameter and an indication of the user action that occurred.

---

## Configuring Link Tracking in Tagged Pages

### Problem

You want to configure which links are tracked on the tagged pages.

### Solution

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 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 but your site also includes links that direct the user to another department or sub-site 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 sub-site'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 but 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 105](#).

*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 105](#)
- ["Tracking Links By ID" on page 105](#)
- ["Tracking Links By Name" on page 106](#)
- ["Tracking Links By Other Attributes" on page 106](#)

## Tasks

### 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 and 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;
    }
}
```

---

## Running a Page-Tagging ETL Job

### Problem

You want to process the data collected by a clickstream collection server.

### Solution

You can process the job in the page tagging job template. Unlike other template job processing, the page tagging template uses two Clickstream Parse transformations to extract the tagged data. The following overview shows the steps that are executed.

1. Clickstream Log: Reads in the tagged data from the raw tagged Web log.
2. Checkpoint for Clickstream Log.
3. Parse Tagged Data Items: This step is responsible for extracting all tagged data elements and for generating output ready for the subsequent Clickstream Parse.
4. Checkpoint for Parse Tagged Data Items.
5. Parse: This step is responsible for processing the data from the original requested file.
6. Checkpoint for Clickstream Parse.
7. Clickstream Sessionize: Sessions the data as normal and includes the tagged data elements extracted in the first Clickstream Parse transformation.
8. Checkpoint for Clickstream Sessionize.

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 page tagging default job:

- [“Prepare the Job” on page 108](#)
- [“Run the Job and Examine the Output” on page 108](#)

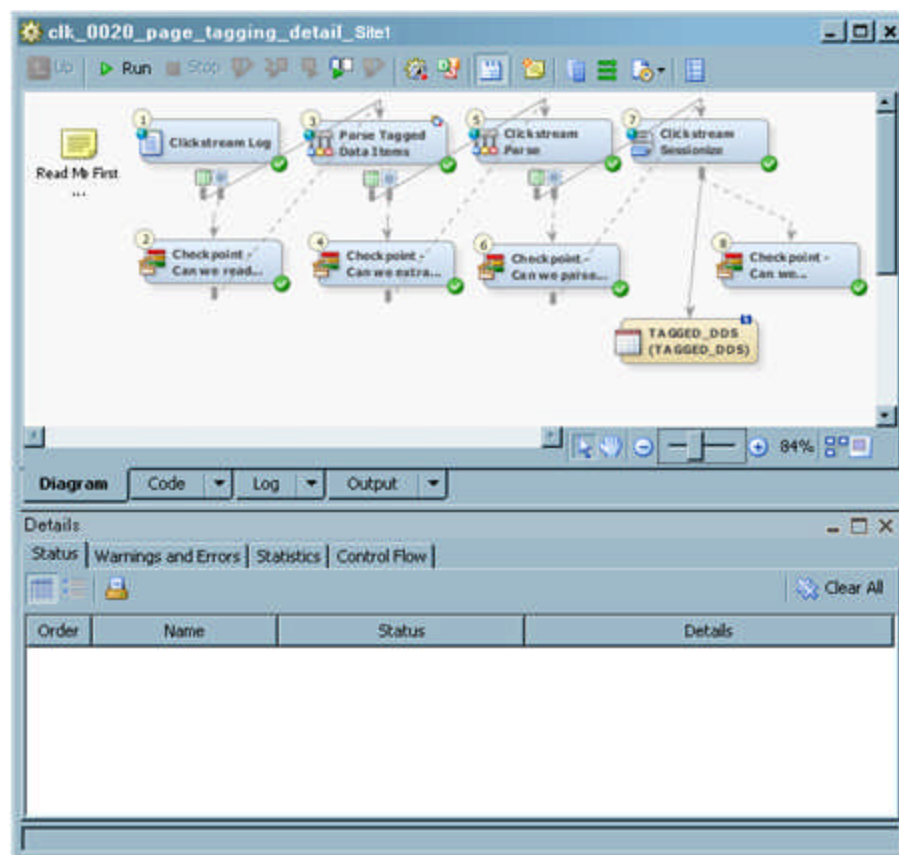
## Tasks

### Prepare the Job

If you have not done so already, you should run a copy of the setup job for the page tagging template, which is named `clk_0010_setup_page_tagging`. When you actually process the data, you should copy and rename the page tagging template job before you run it. For example, you might run a job named `clk_0020_page_tagging_detail_Site1` job. Renaming a copy of the job ensures that you keep the original template job and retain access to its default values. (See “Copying the Page Tagging Template” on page 93.)

The following display shows a sample renamed template job.

**Display 9.1** Copied Page Tagging Template Job



### Run the Job and Examine the Output

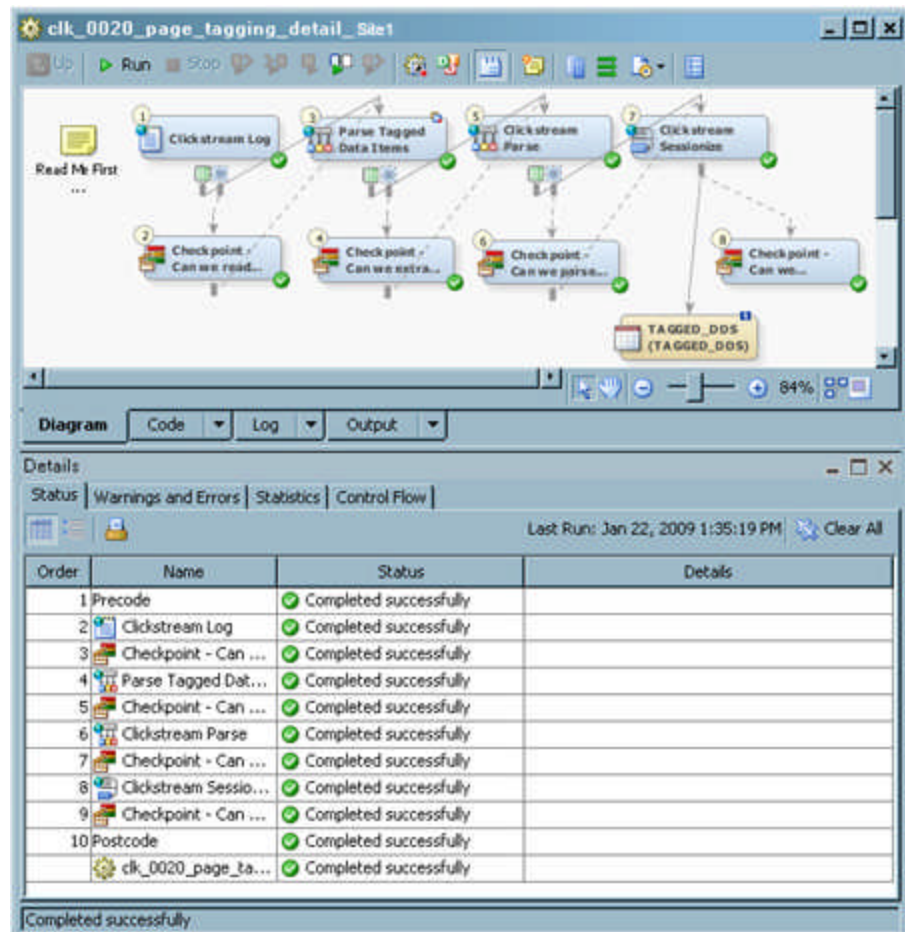
Perform the following steps to run the page tagging job and examine its output:

1. Open the job.



The following display shows a successfully completed job.

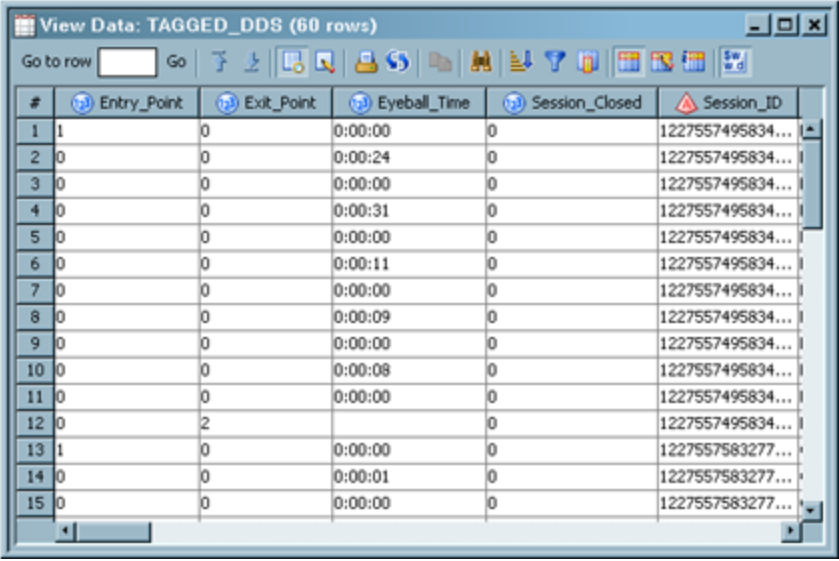
**Display 9.2** Completed Page Tagging Template Job



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

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

**Display 9.3** Page Tagging Output



The screenshot shows a software window titled "View Data: TAGGED\_DDS (60 rows)". It contains a table with 6 columns: #, Entry\_Point, Exit\_Point, Eyeball\_Time, Session\_Closed, and Session\_ID. The table displays 15 rows of data. The first 12 rows have Session\_IDs starting with "1227557495834...", while the last three rows (13-15) have Session\_IDs starting with "1227557583277...".

#	Entry_Point	Exit_Point	Eyeball_Time	Session_Closed	Session_ID
1	1	0	0:00:00	0	1227557495834...
2	0	0	0:00:24	0	1227557495834...
3	0	0	0:00:00	0	1227557495834...
4	0	0	0:00:31	0	1227557495834...
5	0	0	0:00:00	0	1227557495834...
6	0	0	0:00:11	0	1227557495834...
7	0	0	0:00:00	0	1227557495834...
8	0	0	0:00:09	0	1227557495834...
9	0	0	0:00:00	0	1227557495834...
10	0	0	0:00:08	0	1227557495834...
11	0	0	0:00:00	0	1227557495834...
12	0	2		0	1227557495834...
13	1	0	0:00:00	0	1227557583277...
14	0	0	0:00:01	0	1227557583277...
15	0	0	0:00:00	0	1227557583277...

## Appendix 1

# Clickstream Parse Input and Output Columns

### 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 A1.1** Clickstream Parse Input 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	32760	\$32760.
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	8	\$8.
CLK_cs_Referrer	Specifies the full URL and any query parameters from the referring page.	Referrer	1024	\$1024.
CLK_cs_URI_Query	Specifies the query string that is passed to the URL.	Query Sting	1024	\$1024.

Column Name	Description	Label	Length	SAS Format
CLK_cs_URI_Stem	Specifies the URI, which is the URL, but without the http://www.domain.com/ field.	Requested File	1024	\$1024.
CLK_cs_UserAgent	Specifies the string that identifies the user's browser, which the user's browser sends.	User Agent	160	\$160.
CLK_cs_Username	Specifies the user name that the client used for authentication, if applicable.	Username	32	\$32.
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.
CLK_sc_SubStatus	Specifies the secondary status that is returned by some Web servers.	Sub Status	8	4.

Column Name	Description	Label	Length	SAS Format
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 maps the Parse input columns to a set of Parse output columns. The following table lists the metadata for the Clickstream Parse output columns.

**Table A1.2** 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	40	\$40.
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.
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	32760	\$32760.

Column Name	Description	Completion Method	Label	Length	SAS Format
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	128	\$128.
Method	Specifies the method that is used to submit the request (for example, POST or GET).	Pass-Through CLK_cs_Method	Method	8	\$8.
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	40	\$40.
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	1024	\$1024.
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	1024	\$1024.
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	128	\$128.
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.

Column Name	Description	Completion Method	Label	Length	SAS Format
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	1024	\$1024.
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	1024	\$1024.
Requested_File	Specifies the requested file.	Pass-Through CLK_cs_URI_Stem	Requested_File	1024	\$1024.
Server	Specifies the physical computer name that the Web server runs on, such as <b>CLK_s_ComputerName</b> .	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 <b>CLK_s_Port</b> .	Pass-Through CLK_s_Port	Server Port	8	\$8.
Sitename	Specifies the name of the virtual Web site, such as <b>CLK_s_SiteName</b> .	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.
SubStatus	Specifies the secondary status that is returned by some Web servers.	Pass-Through CLK_sc_SubStatus	Sub Status	8	4.

Column Name	Description	Completion Method	Label	Length	SAS Format
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	160	\$160.
Username	Specifies the user name that the client sends to the server for authentication, if applicable.	Pass-Through CLK_cs_Username	Username	32	\$32.
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.



# Index

---

## A

Apache HTTP Server [93](#)

## B

backups [7, 30](#)

Basic (Multiple) Web Log Template Job  
[60, 76](#)

combining groups [65, 79](#)

copying the Basic (Multiple) Web Log  
Template folder [70, 83](#)

creating detail and generating output [69, 82](#)

preparing data and parameter values [61, 76](#)

propagation of columns [60](#)

recognizing, parsing, and grouping data  
[63, 78](#)

running [71, 84](#)

sessionizing [67, 81](#)

basic (single) Web log template [6, 35](#)

copying [38](#)

basic multiple web log template [57](#)

Build Loop Parameters transformation [62, 66, 77, 80](#)

## C

Checkpoint transformations

in Basic (Multiple) Web Log Template  
Job [64, 68, 78, 82](#)

in Single Log Template Job [36, 37](#)

in Subsite Template Job [44, 46, 49](#)

CLICKRC macro variable

resetting [7](#)

clickstream collection servers [92](#)

preparing [93](#)

Clickstream Combine Groups  
transformation [58](#)

Clickstream Create Detail transformation  
[5, 69, 83](#)

Clickstream Create Groups transformation  
[5, 66, 80](#)

clickstream data, defined [2](#)

clickstream jobs

*See also* [jobs](#)

example [3](#)

Clickstream Log transformation

function [9](#)

in Basic (Multiple) Web Log Template  
Job [64](#)

in Multiple Log Template Job [78](#)

in Single Log Template Job [36](#)

in Subsite Template Job [44](#)

maintaining log types [11](#)

managing user columns [13](#)

specifying log options [14](#)

specifying path to Web log [10](#)

clickstream parameters [21](#)

clickstream parse rules [23](#)

Clickstream Parse transformation

applying clickstream parse rules [22](#)

extracting data from clickstream  
parameters [21](#)

functions [16](#)

handling non-human visitors [17](#)

identifying incoming columns [18](#)

in Basic (Multiple) Web Log Template  
Job [64](#)

in Multiple Log Template Job [79](#)

in Single Log Template Job [37](#)

in Subsite Template Job [45, 46, 49](#)

input columns [111](#)

maintaining user columns [19](#)

managing output table columns [25](#)

managing the visitor ID [24](#)

optimizing a sort [17](#)

output columns [113](#)

setting the hold buffer size option [17](#)

setting visitor ID values [18](#)

specifying parse options [25](#)

Clickstream Sessionize transformation

- backing up PERMLIB library 30
- columns generated by 28
- function 27
- in Basic (Multiple) Web Log Template Job 82
- in Multiple Log Template Job 68
- in Single Log Template Job 37
- in Subsite Template Job 46, 49
- managing non-human visitor detection 31
- managing PERMLIB library content 30
- spanning Web logs 32
- specifying options 33
- visitor ID completion 30
- Clickstream Setup transformation 5
- columns
  - generated by Clickstream Sessionize transformation 28
  - input for Clickstream Parse transformation 111
  - maintaining mapping 18
  - maintaining user columns 13, 19
  - managing output 25
  - output for Clickstream Parse transformation 113
  - propagation in Basic (Multiple) Web Log Template Job 60
- cookies, tracking 101
- crawlers
  - See [non-human visitors](#)

**D**

- debugging page tags 98
- Directory Contents transformation 5, 62, 77

**F**

- Filter - Only properly parsed logs transformation 66, 80
- Filter Failed Jobs transformation 69, 83
- forms, tracking 103

**H**

- hold buffer size option 17

**I**

- input columns for Clickstream Parse transformation 111
- input options for Clickstream Sessionize transformation 33

**J**

- Javascript code for page tagging 94
- jobs
  - Basic (Multiple) Web Log Template Job 60
  - clickstream job example 3
  - Customer Integration Template Job 76
  - running a multiple logs job 71, 84
  - running a page-tagging ETL job 107
  - running a single log job 39
  - running a subsite job 54
  - Single Log Template Job 36
  - Subsite Template Job 43

**L**

- libraries
  - backing up PERMLIB library 30
  - managing PERMLIB content 30
- links, tracking 102
- log jobs
  - See *also* [jobs](#)
  - running single 39
- log options 14
- log type definitions 11
- Loop End transformation 65, 68, 79, 82
- Loop transformation 64, 67, 78, 81

**M**

- mapping columns 18
- meta tags, tracking 101
- multiple log template 6
- Multiple Log Template folder, copying 70, 83
- Multiple Log Template Job
  - process flow 58

**N**

- NCSA Common Combined Log Format (CLFE) 11
- NHV
  - See [non-human visitors](#)
- non-human visitors
  - handling in the Clickstream Parse transformation 17
  - managing detection in the Clickstream Sessionize transformation 31

**O**

- options
  - for Clickstream Sessionize transformation 33
  - log 14

- parse 25
- output columns for Clickstream Parse transformation 113
- output tables
  - backing up 7
  - managing columns 25

## P

- page tagging 92
  - customizing tags 98
  - debug mode 98
  - inserting full page tags 96
  - inserting minimal page tags 95
  - Javascript code 94
  - predefined data elements 98
  - running an ETL job 107
  - security issues 92
  - tracking cookies 101
  - tracking forms 103
  - tracking links 102
  - tracking meta tags 101
  - tracking page loads 98
  - tracking rich Internet applications 104
  - user-defined elements for 100
- page tagging template 6, 93
  - copying Page Tagging Template folder 93
  - processing a job 107
- parameters 21
- parse options 25
- performance, optimizing 17
- PERMLIB library
  - backing up 30
  - managing content 30
- pingers
  - See [non-human visitors](#)
- predefined data elements for page tagging 98
- prerequisites for SAS Data Surveyor for Clickstream Data 2

## R

- rich Internet applications, tracking 104
- robots
  - See [non-human visitors](#)

## S

- SAS Data Surveyor for Clickstream Data
  - overview 2
  - prerequisites 2
- SAS Tag Data Format 11
- security for page tagging 93
- Set Output Library transformation 62, 77

- Set Sessionize Output Library Locations transformation 66, 80
- Single Log Template Job 36
  - creating sessions and generating output 37
  - loading and preparing data 36
  - parsing data 37
- SORTSIZE option 17
- spiders
  - See [non-human visitors](#)
- Sub Site Templates folder, copying 50
- subsite flow segments
  - adding 51
  - deleting 53
  - modifying 53
- subsite template 6, 50
- Subsite Template Job 43
  - copying the Sub Site Templates folder 50
  - generating data from site-wide data 48
  - generating subsite sessions 45
  - loading data and applying global rules 44
  - managing subsite flow segments 51
  - running 54
- Sun iPlanet Log Format 11

## T

- table options for Clickstream Sessionize transformation 33
- tables, output
  - See [output tables](#)
- tagging Web pages
  - See [page tagging](#)
- template column metadata 6
- templates 5
  - basic (single) Web log 6, 35
  - copying basic (single) Web log 38
  - multiple log 6, 57
  - page tagging 6, 93
  - subsite 6, 50
- transformations 4
  - Build Loop Parameters 62, 66, 77, 80
  - Checkpoint in Basic (Multiple) Web Log Template Job 64, 68
  - Checkpoint in Multiple Log Template Job 78, 82
  - Checkpoint in Single Log Template Job 36, 37
  - Checkpoint in Subsite Template Job 44, 46, 49
  - Clickstream Combine Groups 58
  - Clickstream Create Detail 5, 69, 83
  - Clickstream Create Groups 5, 66, 80
  - Clickstream Log 9, 36, 44, 64, 78

- Clickstream Parse [16](#), [37](#), [64](#), [79](#)
- Clickstream Parse - ALL [49](#)
- Clickstream Parse - GEN [47](#)
- Clickstream Parse - Global Rules [45](#)
- Clickstream Parse - PRD [46](#)
- Clickstream Parse - SVCS [46](#)
- Clickstream Sessionize [37](#), [68](#), [82](#)
- Clickstream Sessionize - ALL [49](#)
- Clickstream Sessionize - GEN [47](#)
- Clickstream Sessionize - PRD [46](#)
- Clickstream Setup [5](#)
- Directory Contents [5](#), [62](#), [77](#)
- Filter - Only properly parsed logs [66](#), [80](#)
- Filter Failed Jobs [69](#), [83](#)
- Loop [64](#), [67](#), [78](#), [81](#)
- Loop End [65](#), [68](#), [79](#), [82](#)
- Set Output Library [62](#), [77](#)
- Set Sessionize Output Library Locations [66](#), [80](#)
- tuning options for Clickstream Sessionize transformation [33](#)

## U

- user columns

- maintaining in Clickstream Parse transformation [19](#)
- managing in Clickstream Log transformation [13](#)
- selecting as visitor ID [24](#)

## V

- visitor IDs
  - completion in Clickstream Sessionize transformation [30](#)
  - managing [24](#)
  - selecting user columns as [24](#)
  - setting values [18](#)

## W

- W3C Extended Log Format (ELF) [12](#)
- Web logs
  - limitations of standard [92](#)
  - spanning in Clickstream Sessionize transformation [32](#)
  - specifying path to [10](#)

---

## Your Turn

We welcome your feedback.

- If you have comments about this book, please send them to **`yourturn@sas.com`**. Include the full title and page numbers (if applicable).
- If you have comments about the software, please send them to **`suggest@sas.com`**.



# SAS® Publishing Delivers!

Whether you are new to the work force or an experienced professional, you need to distinguish yourself in this rapidly changing and competitive job market. SAS® Publishing provides you with a wide range of resources to help you set yourself apart. Visit us online at [support.sas.com/bookstore](http://support.sas.com/bookstore).

## SAS® Press

Need to learn the basics? Struggling with a programming problem? You'll find the expert answers that you need in example-rich books from SAS Press. Written by experienced SAS professionals from around the world, SAS Press books deliver real-world insights on a broad range of topics for all skill levels.

**[support.sas.com/saspress](http://support.sas.com/saspress)**

## SAS® Documentation

To successfully implement applications using SAS software, companies in every industry and on every continent all turn to the one source for accurate, timely, and reliable information: SAS documentation. We currently produce the following types of reference documentation to improve your work experience:

- Online help that is built into the software.
- Tutorials that are integrated into the product.
- Reference documentation delivered in HTML and PDF – **free** on the Web.
- Hard-copy books.

**[support.sas.com/publishing](http://support.sas.com/publishing)**

## SAS® Publishing News

Subscribe to SAS Publishing News to receive up-to-date information about all new SAS titles, author podcasts, and new Web site features via e-mail. Complete instructions on how to subscribe, as well as access to past issues, are available at our Web site.

**[support.sas.com/spn](http://support.sas.com/spn)**



**THE  
POWER  
TO KNOW®**

