CISCO SERVICE CONTROL SOLUTION GUIDE

Cisco Service Control Online Advertising Solution Guide:

Behavioral Profile Creation Using RDRs

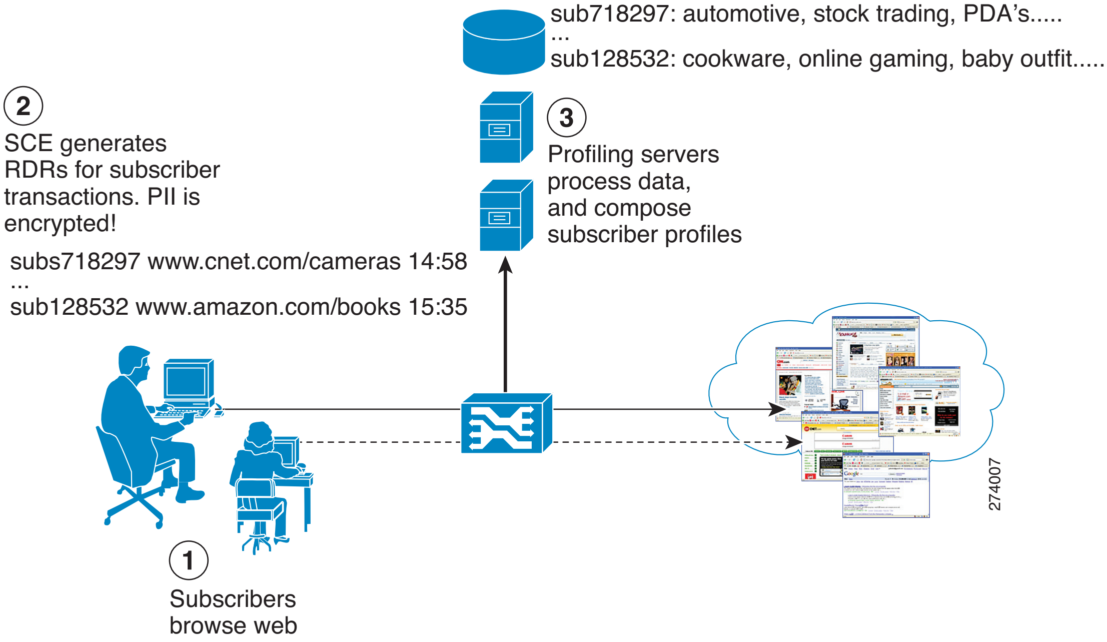
Revised: January, 2009, OL-17778-02

1 Overview

Online Behavioral Targeting is an online advertising approach that is based on presenting users with ads based on their interests, as deduced through monitoring their web browsing. The SCE platform can enable this based on an analysis of subscriber online usage patterns.

Such behavioral targeting does not require the analysis of each and every HTTP request on the line, as this would result in a lot of excess information. The SCE platform performs the first level of analysis in the behavioral targeting chain by inspecting the user browsing sessions, detecting the particular requests that are triggered by the actual user browsing (these events are termed ClickStream) and generating RDRs that contain a digest of these events. In order not to compromise subscriber privacy, the RDRs can be configured not to include any Personally Identifiable Information (PII), in which case all elements containing PII are hashed. The RDRs are typically received by an entity that analyzes the nature of usage and creates a profile of the subscriber to be used later for targeting. The way the greater solution works is outside the scope of this document.

Figure 1 .High Level Overview of an RDR-based Behavioral Targeting Solution


sub718297: automotive, stock trading, PDA's.....
...
sub128532: cookware, online gaming, baby outfit.....
2
SCE generates RDRs for subscriber transactions. PII is encrypted!
3
Profiling servers process data, and compose subscriber profiles
subs718297 www.cnet.com/cameras 14:58
...
sub128532 www.amazon.com/books 15:35
1
Subscribers browse web
274007

ClickStream detection is a fundamental capability of the solution, as it can detect which specific requests, out of the enormous number of HTTP requests generated throughout the subscriber web activity, were actually triggered by the subscriber browsing the web. When a subscriber clicks a link, or types a URL into the browser address bar, an http request is generated to fetch this URL. Typically, an html page is returned, which constitutes the outline of the contents requested. For the browser to be able to render this page, it must download multiple objects (tens or sometimes around a hundred for a single page viewed), which in turn results in multiple http requests for obtaining these objects.

To be able to conduct behavioral targeting, it is typically sufficient to understand what the user was trying to do (represented by the initial request, such as biz.publisher.com/ap/081120/world_markets.html -->"global markets"), rather than looking at each and every object downloaded as a secondary result of such a request (such as http://ads.adnetwork.com/a/a/in/interbroke/300x250_yah.jpg --> broker).

ClickStream detection makes exactly this distinction, allowing the number of requests to be analyzed to be greatly reduced, which is necessary in order to enable a scalable analysis solution.

The information that is collected per such transaction is exported from the SCE using Extended Transaction Usage RDRs, which include information on the transaction that has been performed, in addition to information on the subscriber that performed the transaction.

PII, such as subscriber ID and IP address, is protected by hashing this information with the use of preconfigured "salt". In this way, the subscriber record can be matched only by another system configured with the same salt, in which case it has the original PII to match against. Salt is configurable to the SCE platform.

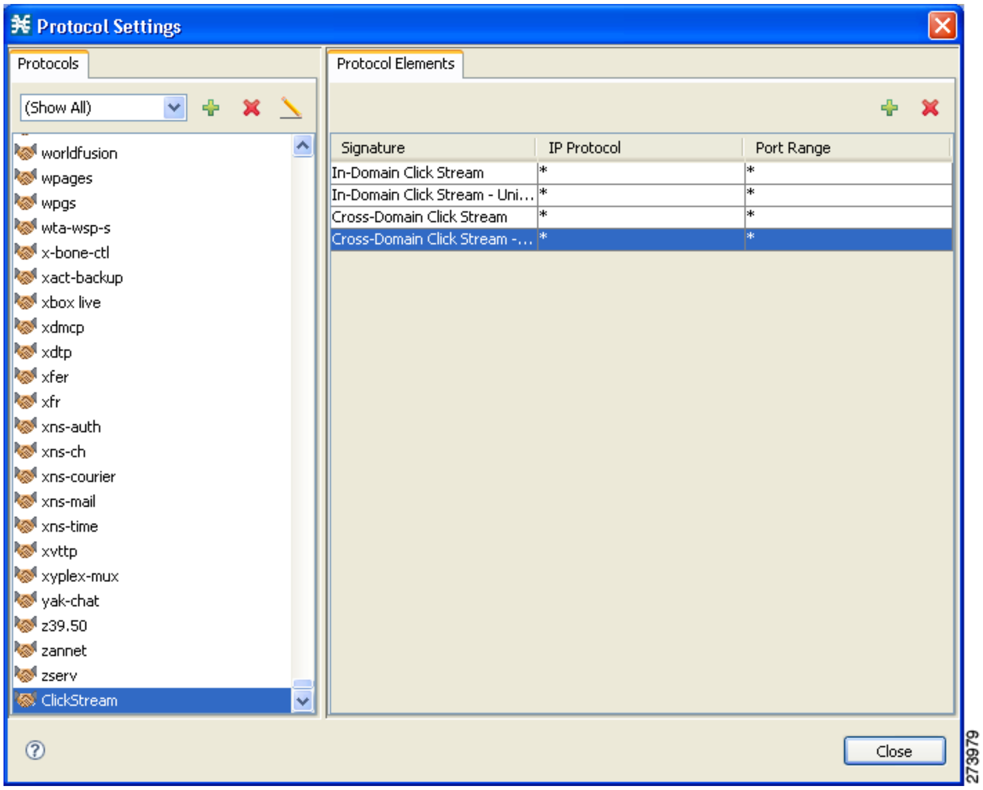
2 Configuring Behavioral Targeting Support: Highlights

This section provides the highlights of configuring the main components of behavioral targeting on the SCE platform. For more detailed configuration directions, see Step by Step Guide: Configuring an SCE Platform for Exporting Behavioral Targeting Information, page 7

Creating a ClickStream Service

ClickStream signatures are mapped by default to the HTTP Browsing protocol and consequently to the browsing service. In order to be able to report on them separately, you will first have to move them to a protocol of their own, and assign this protocol to a service of its own.

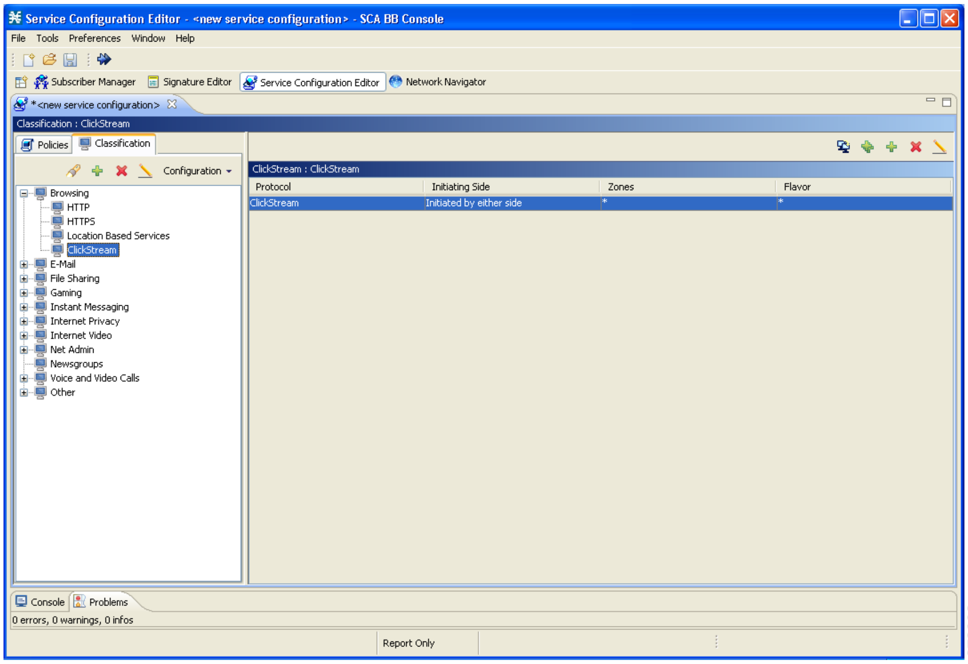
Figure 2 Configuring the ClickStream Protocol



- worldfusion
- wpages
- wpgs
- wta-wsp-s
- x-bone-ctl
- xact-backup
- xbox live
- xdmcp
- xdtp
- xfer
- xfr
- xns-auth
- xns-ch
- xns-courier
- xns-mail
- xns-time
- xvttp
- xyplex-mux
- yak-chat
- z39.50
- zannet
- zserv
- ClickStream

Signature	IP Protocol	Port Range
In-Domain Click Stream	*	*
In-Domain Click Stream - Uni…	*	*
Cross-Domain Click Stream	*	*
Cross-Domain Click Stream -…	*	*

Figure 3 Configuring the ClickStream Service



Protocol	Initiating Side	Zones	Flavor
ClickStream	Initiated by either side	*	*

0 errors, 0 warnings, 0 infos

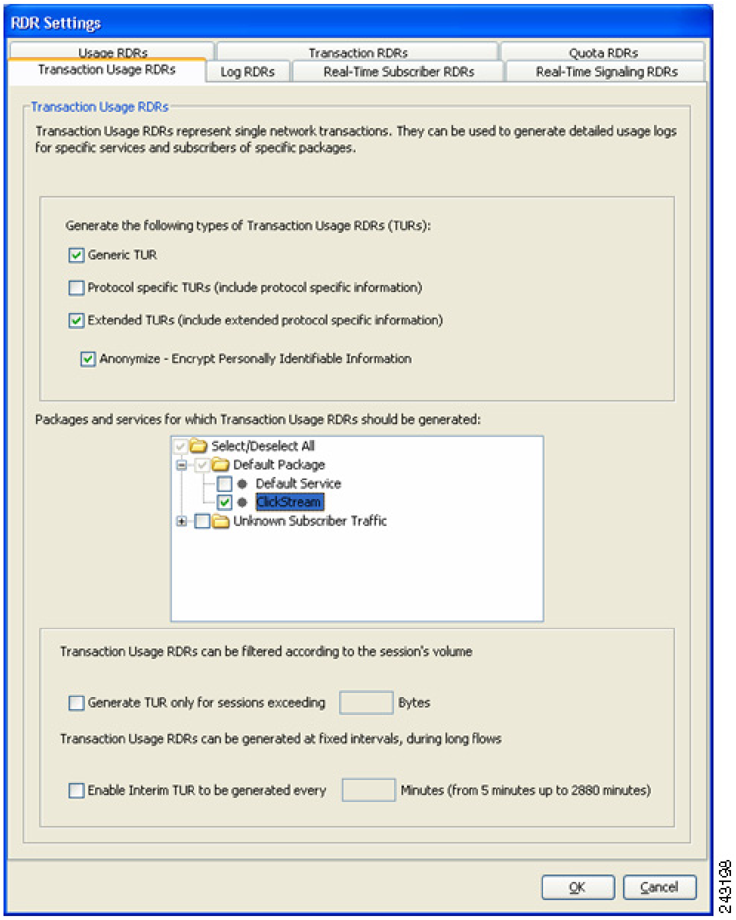
Configuring Extended Transaction Usage RDRs (Extended TURs)

Extended TURs contain extra protocol specific information (see Anonymized HTTP Transaction Usage RDR, page 16). These RDRs are configured from the **Transaction Usage RDRs** tab of the RDR Settings dialog of the SCA BB console.

You enable these RDRs explicitly by checking the appropriate checkbox, and choosing the packages you want them generated for (you only need to enable these RDRs for the ClickStream service in each package).

Optionally, these RDRs can be anonymized, in which case all Personally Identifiable Information (PII) in the RDR will be hashed using MD5. Hashing will be done using the salt configured on the SCE platform.

Figure 4 Transaction Usage RDRs Tab



Transaction Usage RDRs

Transaction Usage RDRs represent single network transactions. They can be used to generate detailed usage logs for specific services and subscribers of specific packages.

Generate the following types of Transaction Usage RDRs (TURs):

☑ Generic TUR

□ Protocol specific TURs (include protocol specific information)

☑ Extended TURs (include extended protocol specific information)

☑ Anonymize - Encrypt Personally Identifiable Information

Packages and services for which Transaction Usage RDRs should be generated:

☑ Select/Deselect All
☑ Default Package
□ Default Service
☑ ClickStream
□ Unknown Subscriber Traffic

Transaction Usage RDRs can be filtered according to the session's volume

□ Generate TUR only for sessions exceeding ____ Bytes

Transaction Usage RDRs can be generated at fixed intervals, during long flows

□ Enable Interim TUR to be generated every ____ Minutes (from 5 minutes up to 2880 minutes)

Enabling Deep HTTP Inspection

In order to ensure comprehensive detection of the ClickStream events in the traffic stream, it is important to enable deep inspection of HTTP, which configures the SCE platform to analyze and classify all HTTP requests within a single flow.

Some browsers, in conjunction with some web server implementations, will use the same TCP flow to carry multiple requests triggered by clicks that are targeting the same host. Such events will not be detected if the classification is only done at the beginning of the flow (which is the default for SCA BB).

To enable deep HTTP inspection, in the SCA BB Console Service Configuration Editor, go to:

Configuration>System Settings>Advanced Options tab**>Advanced Service Configuration Options...**

Note Enabling deep HTTP inspection is expected to impact the SCE performance due to the excessive processing associated with it; the actual figure depending on the amount and on the nature of HTTP traffic. It is recommended that you monitor SCE platform performance when enabling this capability.

Configuring MD5 Salt to the SCE Platform

When the **Anonymize** option is selected, Extended TURs are generated with all personally identifying fields (subscriber ID and subscriber IP address) hashed using MD5.

The SCE platform applies a 'salt' to the Personally Identifying Field prior to hashing it. The salt is 128 bits (16 bytes) long, and it is configured to the SCE platform in four separate 4-byte arguments represented in HEX. The default value of the salt is 0x12345678 0x12345678 0x12345678 0x12345678.

The salt is configured to the SCE using the following CLI commands with four 4-byte arguments in HEX.

```
SCE#config
SCE#interface linecard 0
SCE(config if)#salt 0x12345678 0x00004321 0xfafafafa 0xafafafaf
```

To return the salt to the default value use the following CLI commands:

```
SCE#config
SCE#interface linecard 0
SCE(config if)#default salt
```

Always make sure to save the running configuration using the following command:

```
SCE# copy running-config startup-config
```

Configuring RDR Routing to the Profiling Server

You can make sure that only ClickStream RDRs are sent to the designated server, even if other RDRs are enabled on the system. This is done by directing the extended HTTP TUR to a separate category, and routing this category to a dedicated server.

This is accomplished in two steps:

1. Direct the HTTP Extended TUR to an exclusive RDR category.

   Map the Extended TUR for HTTP (tag 0xF0F0F53C / 4042323260) to RDR category 2 using the following CLI command:

   ```
   SCE(config)#RDR-formatter rdr-mapping tag-ID 0xF0F0F53C category-number 2
   ```

2. Send the RDRs in this category to the designated server.

   Configure RDR category 2 to the desired destination using the following CLI command:

   ```
   SCE(config)#RDR-formatter destination 10.10.10.10 port 33000 category number 2 priority 100
   ```

You can configure a second destination for the category 2 RDRs to function as a backup destination.

3. Configure a secondary destination for backup as needed.

   Configure RDR category 2 to a secondary destination using the following CLI command:

   ```
   SCE(config)#RDR-formatter destination 10.10.10.11 port 33000 category number 2 priority 90
   ```

4. Save the running configuration.

   ```
   SCE#> copy running-config startup-config
   ```

3 Step by Step Guide: Configuring an SCE Platform for Exporting Behavioral Targeting Information

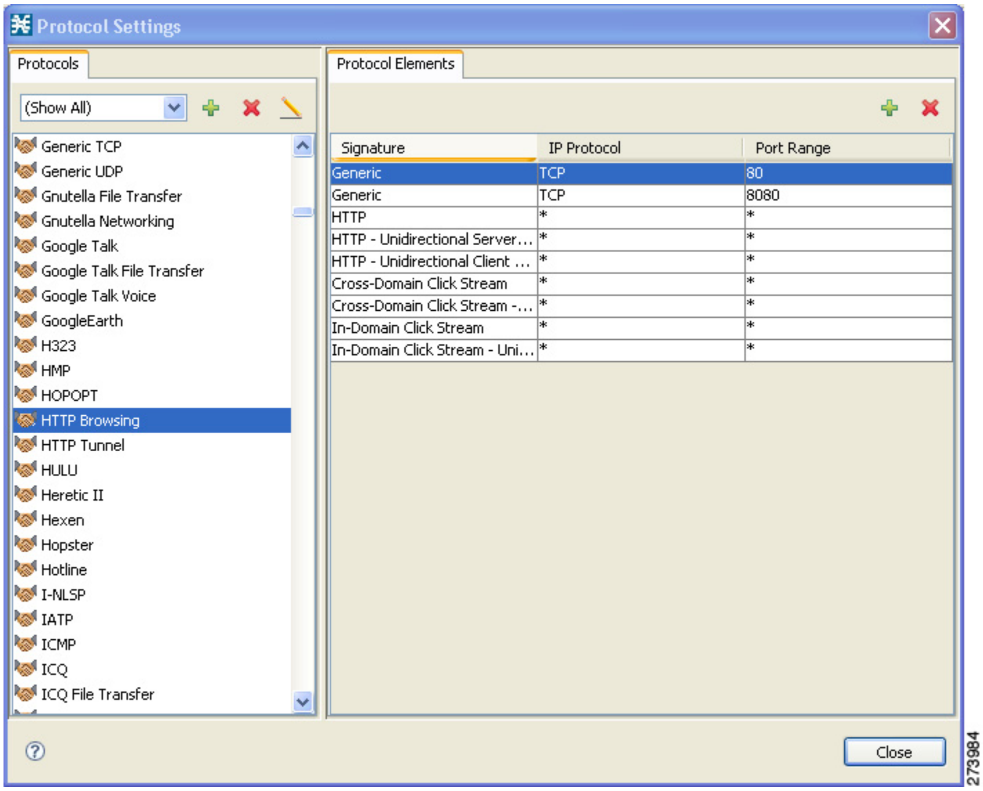
This section explains in detail how to configure an SCE to generate ClickStream RDRs.

Step 1 In the SCA BB Policy Editor, select the **Classification** tab (left pane), click **Configuration,** and select **Protocols**.

Step 2 In the Protocol Settings window, select the **HTTP Browsing** service.

Step 3 In the **Protocol Elements** tab, remove the ClickStream related protocol elements:

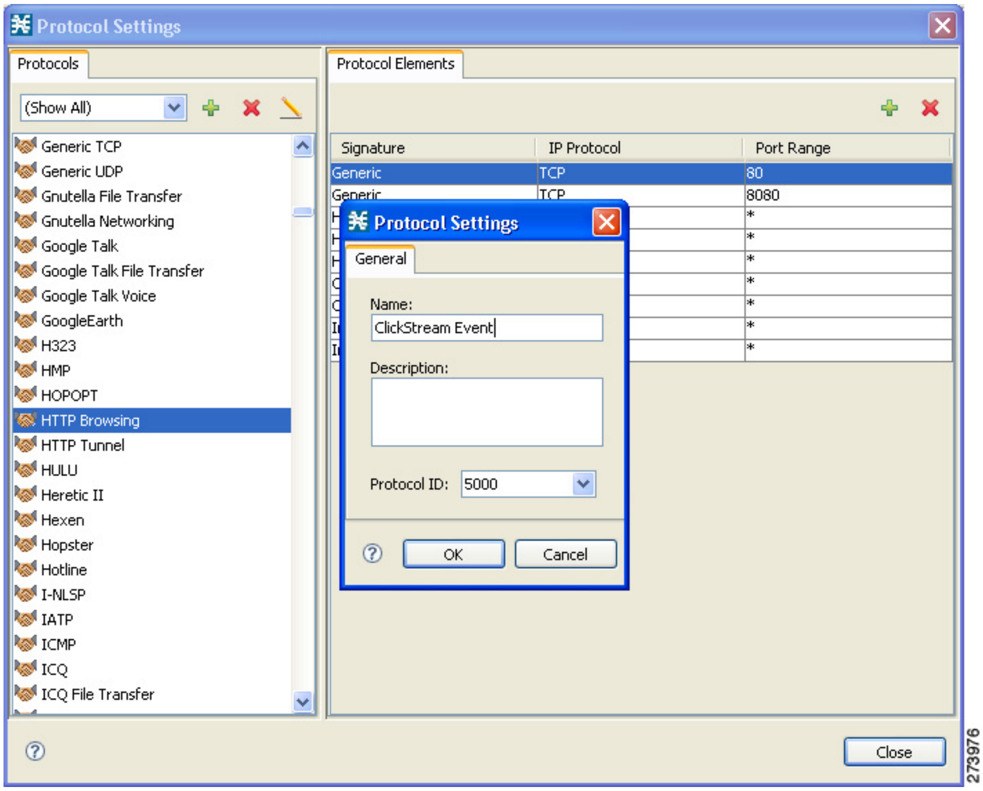
- In-Domain Click Stream
- In-Domain Click Stream - Unidirectional Client Request
- Cross-Domain Click Stream
- Cross-Domain Click Stream - Unidirectional Client Request



Signature	IP Protocol	Port Range
Generic	TCP	80
Generic	TCP	8080
HTTP	*	*
HTTP - Unidirectional Server…	*	*
HTTP - Unidirectional Client …	*	*
Cross-Domain Click Stream	*	*
Cross-Domain Click Stream -…	*	*
In-Domain Click Stream	*	*
In-Domain Click Stream - Uni…	*	*

Step 4 In the Protocol Settings window, on the Protocols tab, click the '+' to add a new protocol.

Step 5 Name the new protocol **ClickStream Event** and click **OK**.



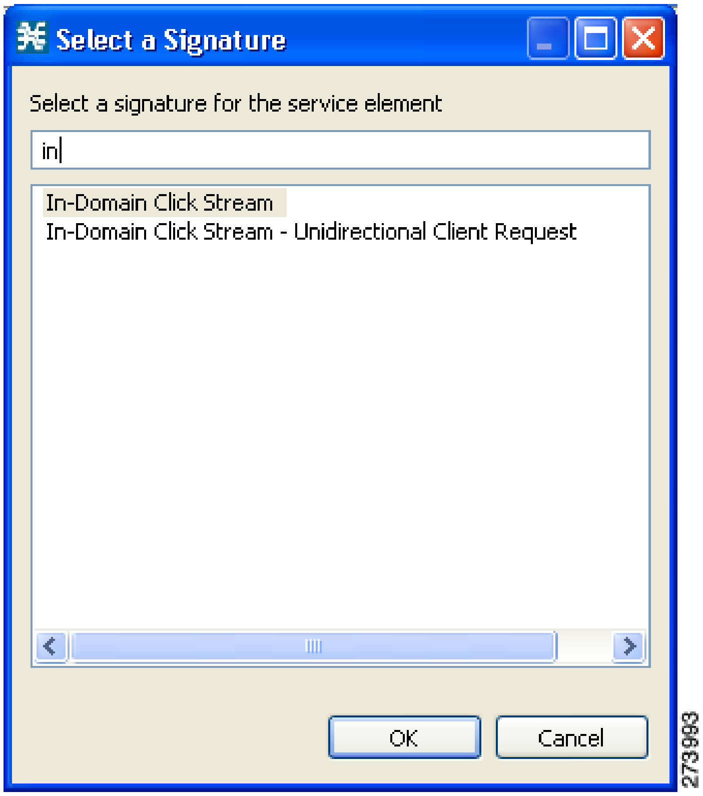
Name: ClickStream Event

Protocol ID: 5000

Step 6 In the Protocol Elements tab, click the '+' to add protocol elements to the ClickStream Protocol.

Step 7 For the new protocol element created., click in the Signature column on the '...' button.

Step 8 On the Select a Signature screen, add the **In-Domain Click Stream** signature and click **OK**.


Select a Signature

Select a signature for the service element

in

In-Domain Click Stream
In-Domain Click Stream - Unidirectional Client Request

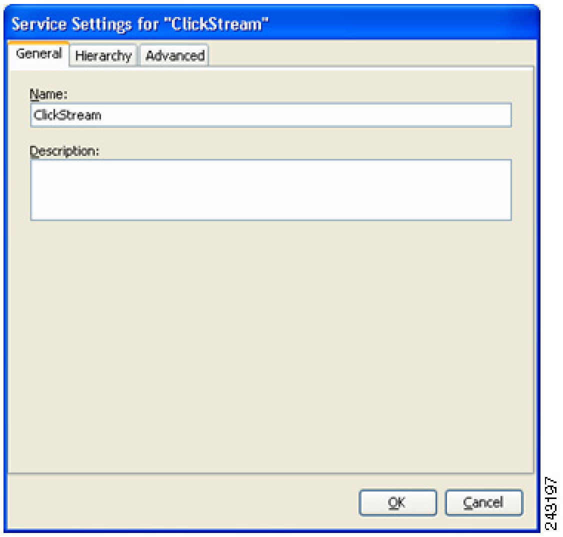
Step 9 Repeat Step 6 through Step 8 for the rest of the ClickStream signatures:

- In-Domain Click Stream - Unidirectional Client Request
- Cross-Domain Click Stream
- Cross-Domain Click Stream - Unidirectional Client Request

Step 10 In the SCA BB Policy Editor, select the Classification tab (left pane), and highlight the Browsing service

Step 11 Click the '+' to add a new service under the Browsing service.

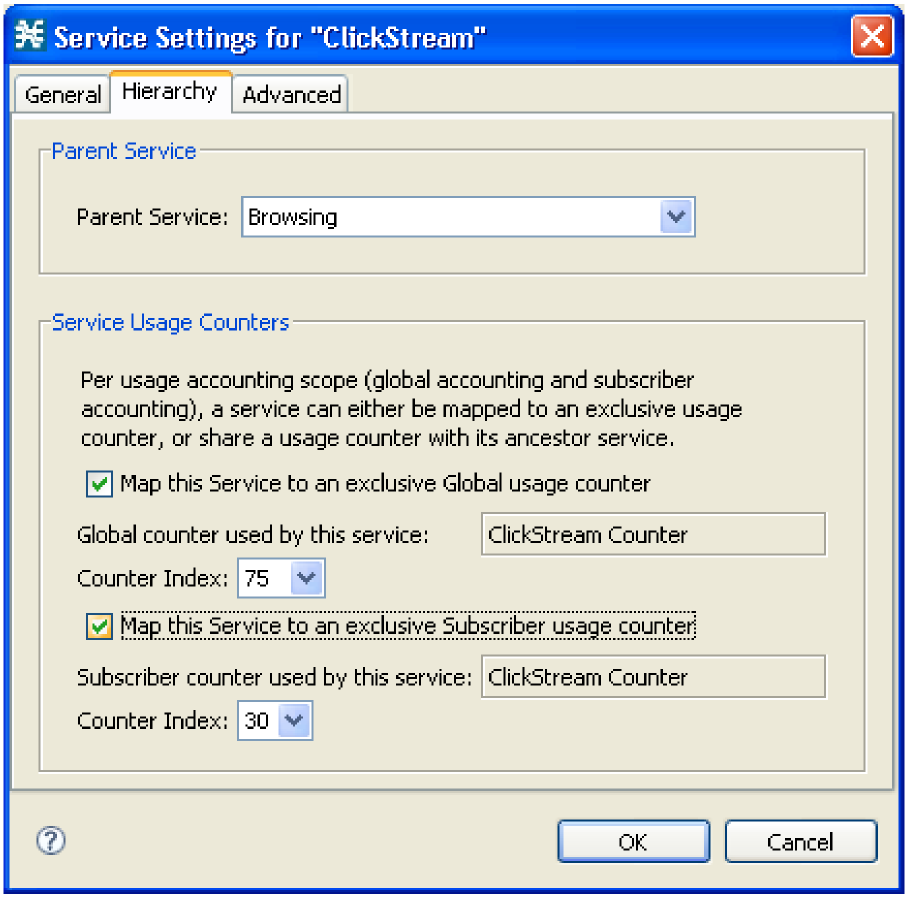
Step 12 Name the service **ClickStream** (or any other name you choose).


Service Settings for "ClickStream"

Name: ClickStream

Description:

Step 13 Click the **Hierarchy** tab and check the two check boxes to add a dedicated service counter to the ClickStream Service. (This is useful if you want to generate reports on the global and per-subscriber ClickStream activity).



Parent Service: Browsing

Per usage accounting scope (global accounting and subscriber accounting), a service can either be mapped to an exclusive usage counter, or share a usage counter with its ancestor service.

☑ Map this Service to an exclusive Global usage counter

Global counter used by this service: ClickStream Counter

Counter Index: 75

☑ Map this Service to an exclusive Subscriber usage counter

Subscriber counter used by this service: ClickStream Counter

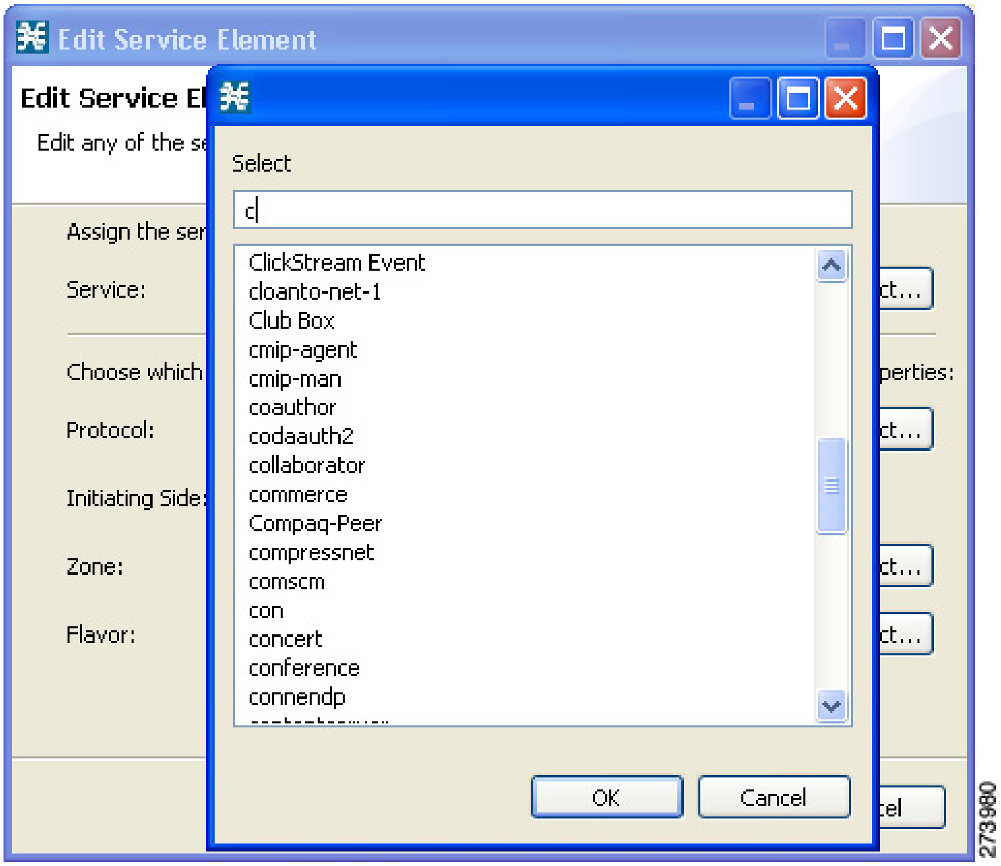
Counter Index: 30

Step 14 Click **OK**.

Step 15 In the right pane, click the '**+**' icon to add a service element.

Step 16 In the dialog that opens, click **Select** next to the Protocol field and select the **ClickStream Event** protocol (or whatever you named your ClickStream protocol) from the list.

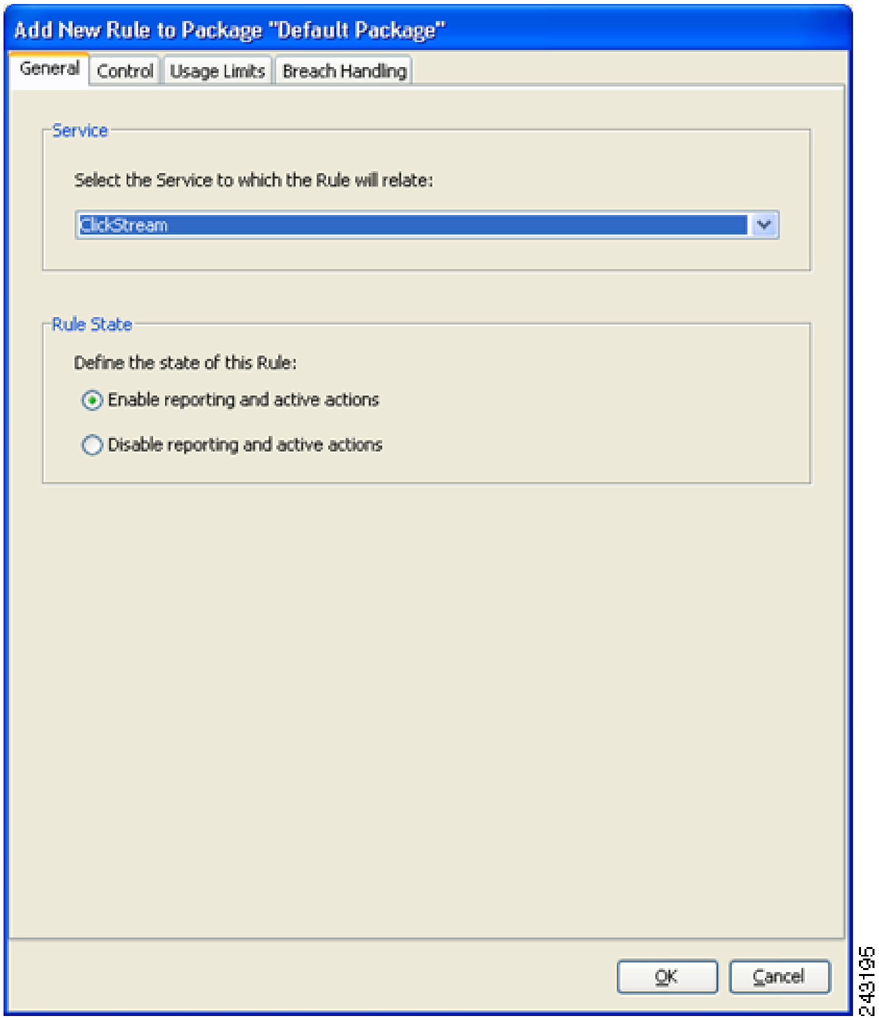
Step 17 Click **OK**.



Step 18 In the SCA BB Policy Editor, select the **Policies** tab (left pane), and then select the package for which to generate the ClickStream RDRs.

Step 19 In the right pane, click the '**+**' icon to add the ClickStream service.

Step 20 In the dialog that opens, select **ClickStream** from the drop-down selection.



Add New Rule to Package "Default Package"

Service

Select the Service to which the Rule will relate: ClickStream

Rule State

Define the state of this Rule:

● Enable reporting and active actions

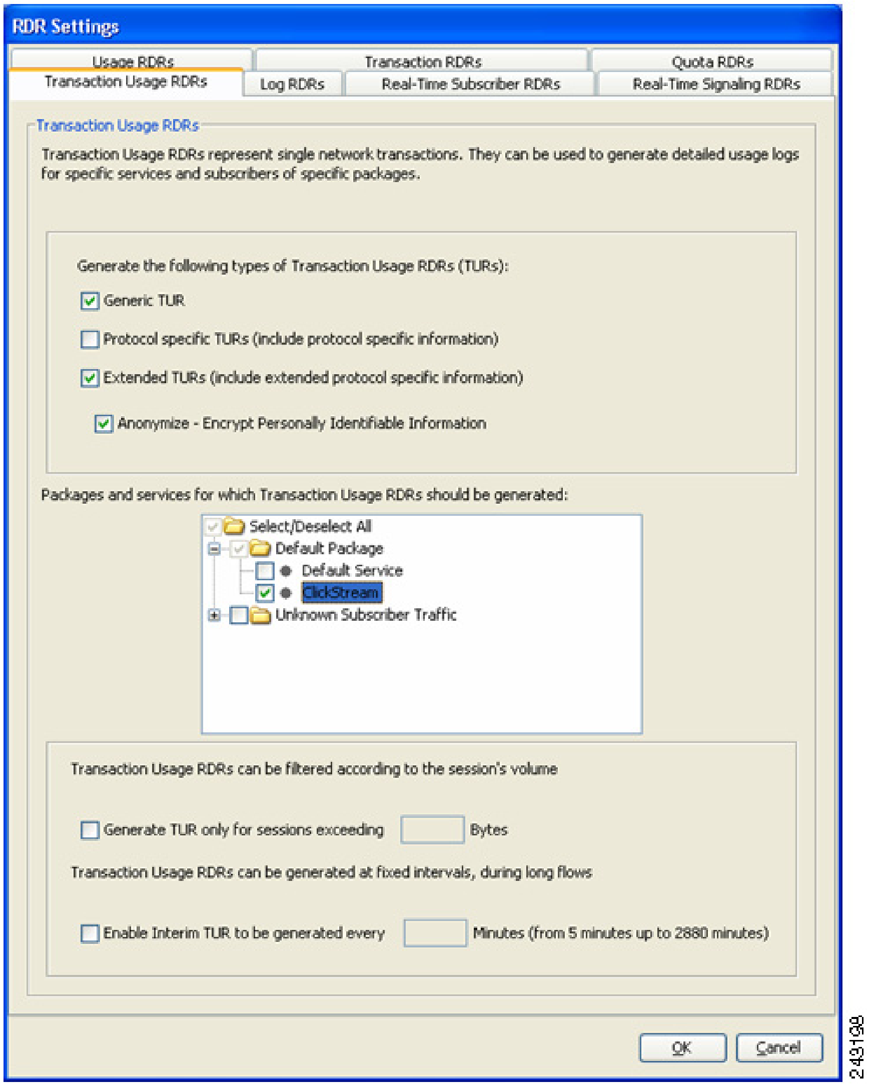
○ Disable reporting and active actions

Step 21 Click **OK**.

Step 22 In the SCA BB Policy Editor, repeat Step 18 through Step 21 for every package for which you want to enable ClickStream RDRs.

Step 23 In the right pane, click on **Configuration** and select **Classification > RDR Settings**.

Step 24 Select the **Transaction Usage RDRs** tab.



Transaction Usage RDRs

Transaction Usage RDRs represent single network transactions. They can be used to generate detailed usage logs for specific services and subscribers of specific packages.

Generate the following types of Transaction Usage RDRs (TURs):

☑ Generic TUR

□ Protocol specific TURs (include protocol specific information)

☑ Extended TURs (include extended protocol specific information)

☑ Anonymize - Encrypt Personally Identifiable Information

Packages and services for which Transaction Usage RDRs should be generated:

- ☑ Select/Deselect All
  - ☑ Default Package
    - □ ● Default Service
    - ☑ ● ClickStream
  - □ Unknown Subscriber Traffic

Transaction Usage RDRs can be filtered according to the session's volume

□ Generate TUR only for sessions exceeding ____ Bytes

Transaction Usage RDRs can be generated at fixed intervals, during long flows

□ Enable Interim TUR to be generated every ____ Minutes (from 5 minutes up to 2880 minutes)

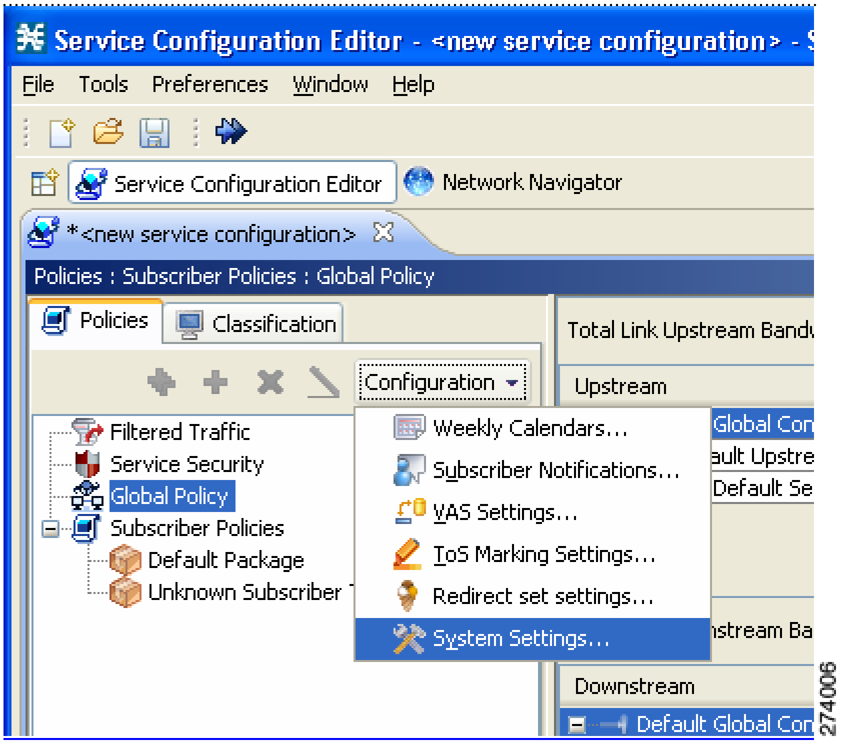
Step 25 Enable Extended TURs .by checking the relevant check box.

Step 26 Optional: Enable anonymization as needed, by checking **Anonymize**.

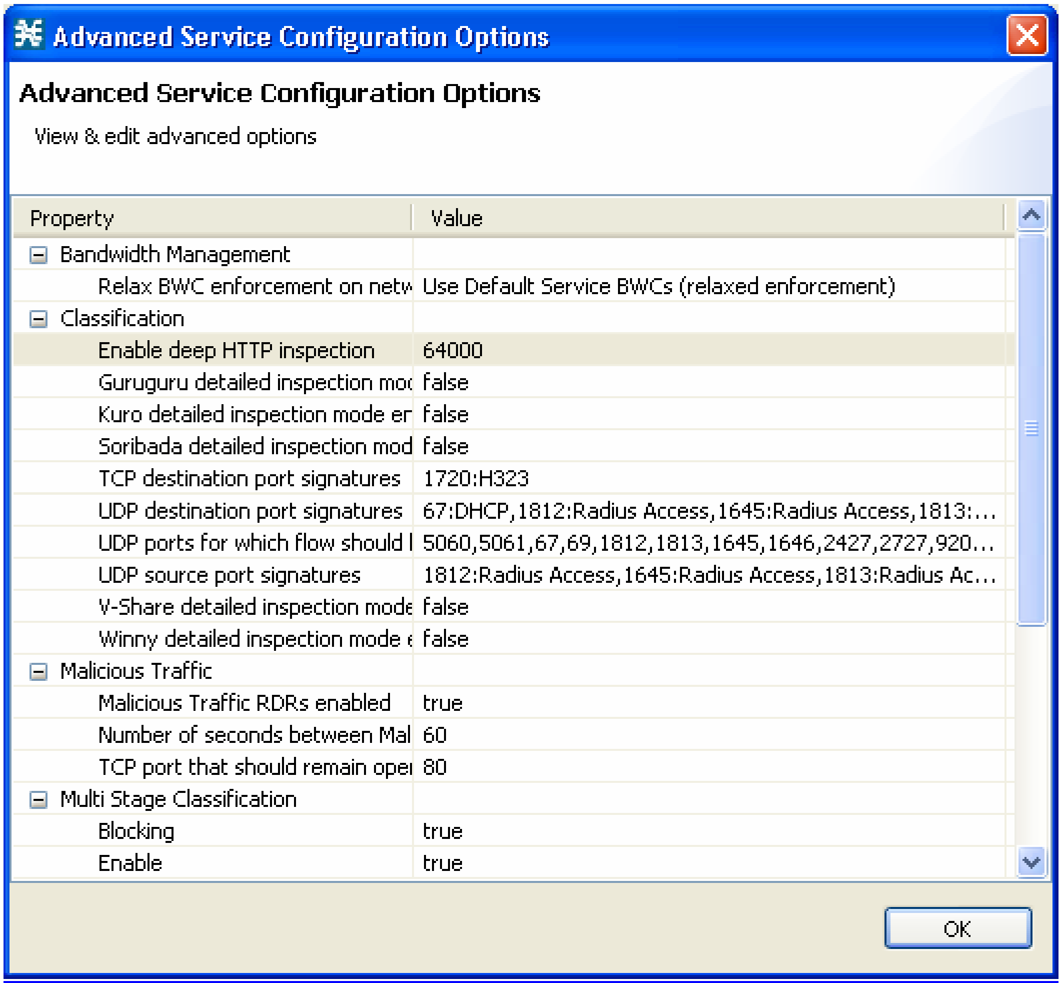
Step 27 Check the ClickStream service on all relevant packages, to enable generating RDRs for this service. Uncheck any other service that you do not want Transaction Usage RDRs (Extended or regular) generated for.

Note Make sure you have already created a specific rule for ClickStream on all relevant packages.

Step 28 Select **Policies > Configuration >System Settings.**



Step 29 In the Advanced Options tab, click **Advanced Service Configuration Options** to enable deep inspection of http flows (beyond the first transaction) by setting the highlighted value to **64000.** This enables the analysis of multiple transactions within a single HTTP flow, which is important for comprehensive detection of ClickStream events.



Advanced Service Configuration Options

View & edit advanced options

Property	Value
⊟ Bandwidth Management	
Relax BWC enforcement on netw	Use Default Service BWCs (relaxed enforcement)
⊟ Classification	
Enable deep HTTP inspection	64000
Guruguru detailed inspection mo(	false
Kuro detailed inspection mode er	false
Soribada detailed inspection mod	false
TCP destination port signatures	1720:H323
UDP destination port signatures	67:DHCP,1812:Radius Access,1645:Radius Access,1813:...
UDP ports for which flow should l	5060,5061,67,69,1812,1813,1645,1646,2427,2727,920...
UDP source port signatures	1812:Radius Access,1645:Radius Access,1813:Radius Ac...
V-Share detailed inspection mode	false
Winny detailed inspection mode (	false
⊟ Malicious Traffic	
Malicious Traffic RDRs enabled	true
Number of seconds between Mal	60
TCP port that should remain ope|	80
⊟ Multi Stage Classification	
Blocking	true
Enable	true

274005

Step 30 Apply the Service Configuration to the SCE platform.

Step 31 (Optional) If anonymization is enabled, configure MD5 salt to the SCE platform (use a 128-bit salt value of your choice).

```
SCE(config if)#>salt 0xfafafafa 0xfafafafa 0xfafafafa 0xfafafafa
```

Step 32 Map the Extended TUR for HTTP (tag 0xF0F0F53C / 4042323260) to RDR category 2:

```
SCE(config)#RDR-formatter rdr-mapping tag-ID 0xF0F0F53C category-number 2
```

Step 33 Direct category 2 to your server of choice:

```
SCE(config)#RDR-formatter destination 10.10.10.10 port 33000 category number 2 priority 99
```

Step 34 Save the configuration:

```
SCE#copy running-config-all startup-config-all
```

4 Anonymized HTTP Transaction Usage RDR

The RDR tag of the Anonymized HTTP Transaction Usage RDR is **F0 F0 F5 3C / 4042323260**

The following table lists the RDR fields and their descriptions.

Table 1 Anonymized HTTP Transaction Usage RDR Fields

RDR Field Name	Type	Description
HASHED_SUBSCRIBER_ID	STRING	The subscriber identification string, passed through hashing algorithm as described below.
PACKAGE_ID	INT16	The ID of the Package assigned to the subscriber whose traffic is being reported.
SERVICE_ID	INT32	The service classification of the reported session.
PROTOCOL_ID	INT16	The unique ID of the protocol associated with the reported session.
SKIPPED_SESSIONS	UINT32	Always 1
SERVER_IP	UINT32	The http server IP. If this is the subscriber IP, this field contains the short-hash of the IP, as described below.
SERVER_PORT	UINT16	The destination port number of the networking session.
HOST	STRING	The Host extracted from the HTTP transaction.
URL	STRING	The URL extracted from the HTTP transaction.
CLIENT_IP	UINT32	The http client IP. If this is the subscriber IP, this field contains the short-hash of the IP, as described below.
CLIENT_PORT	UINT16	The port number of the client side (initiator) of the networking session.
INITIATING_SIDE	INT8	The side of the SCE platform on which the initiator of the transaction resides. • 0-The subscriber side • 1-The network side
REPORT_TIME	UINT32	Ending time stamp of this RDR
MILLISEC_DURATION	UINT32	Duration, in milliseconds, of the transaction reported in this RDR.
TIME_FRAME	INT8	The time frame during which the RDR was generated. (0 to 3)
SESSION_UPSTREAM_VOLUME	UINT32	Upstream volume of the transaction, in bytes. The volume refers to the aggregated upstream volume on both links of all the flows bundled in the transaction.
SESSION_DOWNSTREAM_VOLUME	UINT32	Downstream volume of the transaction, in bytes. The volume refers to the aggregated stream volume on both links of all the flows bundled in the transaction.
SUBSCRIBER_COUNTER_ID	UINT16	Each service is mapped to a counter. There are 32 subscriber usage counters.
GLOBAL_COUNTER_ID	UINT16	Each service is mapped to a counter. There are 64 global usage counters.
PACKAGE_COUNTER_ID	UINT16	Each package is mapped to a counter. There are 1024 package usage counters.
IP_PROTOCOL	UINT8	IP protocol type.
PROTOCOL_SIGNATURE	UINT32	ID of the protocol signature associated with this session.
ZONE_ID	UINT32	ID of the zone associated with this session.

Table 1 Anonymized HTTP Transaction Usage RDR Fields

RDR Field Name	Type	Description
FLAVOR_ID	UINT32	For protocol signatures that have flavors, this field contains the ID of the flavor associated with this session.
FLOW_CLOSE_MODE	UINT8	The reason for the end of flow.
HASHED_SUBSCRIBER_IP	STRING	The subscriber IP, hashed as described below.
USER_AGENT	STRING	The user agent field extracted from the HTTP transaction.
HTTP_REFERER	STRING	The REFERER extracted from the HTTP transaction
HTTP_COOKIE	STRING	The COOKIE extracted from the HTTP transaction.

Further Elaboration on RDR Fields

- HASHED_SUBSCRIBER_ID—The subscriber identification string, introduced through the subscriber management interfaces, passed through hashing algorithm as described below.

  The field is a 32-byte-long string, containing a Hexadecimal notation of the 128 bit hash result.
- PACKAGE_ID—The ID of the Package assigned to the subscriber whose traffic is being reported. An assigned Package ID is an integer value between 0 and maximum_number_of_packages. The value maximum_number_of_packages is reserved for unknown subscribers.
- BREACH_STATE—This field indicates whether the subscriber quota was breached.
  - 0—Not breached
  - 1—Breached
- HASHED_SUBSCRBIER_IP —The IP address of the subscriber side of the reported session, after passing through the hashing algorithm as described below.

  The field is a 32-byte-long string, containing a Hexadecimal notation of the 128 bit hash result.
- CLIENT_PORT—For TCP/UDP-based sessions, the port number of the client side (initiator) of the networking session. For non-TCP/UDP sessions, this field has the value zero.
- CONFIGURED_DURATION—For periodic RDRs, the configured period, in seconds, between successive RDRs.
- END_TIME—Ending time stamp of this RDR. The field is in UNIX time_t format, which is the number of seconds since midnight of 1 January 1970.
- FLAVOR_ID—For protocol signatures that have flavors, this field contains the ID of the flavor associated with this session.
- INITIATING_SIDE—On which side of the SCE platform the initiator of the transaction resides.
  - 0—The subscriber side
  - 1—The network side
- PROTOCOL_ID—This field contains the unique ID of the protocol associated with the reported session.

Note The PROTOCOL_ID will be the Generic IP / Generic TCP / Generic UDP protocol ID value, according to the specific transport protocol of the transaction, unless a more specific protocol definition (such as a signature-based protocol or a port-based protocol), which matches the reported session, is assigned to a service.

- PROTOCOL_SIGNATURE—This field contains the ID of the protocol signature associated with this session.
- REPORT_TIME—Ending time stamp of this RDR. The field is in UNIX time_t format, which is the number of seconds since midnight of 1 January 1970.
- SERVER_IP—Contains the destination IP address of the reported session. (The destination is defined as the server or the listener of the networking session.) The IP address is in a 32-bit binary format.
- SERVER_PORT—For TCP/UDP-based sessions, this field contains the destination port number of the networking session. For non-TCP/UDP sessions, this field contains the IP protocol number of the session flow.

- SERVICE_ID—This field indicates the service classification of the reported session. For example, in the Transaction RDR this field indicates which service was accessed, and in the Breaching RDR this field indicates which service was breached.
- TIME_FRAME—The system supports time-dependent policies, by using different rules for different time frames. This field indicates the time frame during which the RDR was generated. The value can be in the range 0 to 3, indicating which of the four time frames was used.
- ZONE_ID—This field contains the ID of the zone associated with this session.

Note All volumes in RDRs are reported in L3 bytes.

5 Hash Algorithm

Hashing is done using the MD5 hash function, using a 'salt' prepended to the hashed value.

- 'salt' means adding a predefined value before the to-be hashed fields, to avoid simple backtracking of the source argument. The salt currently used by the SCE platform is 128 bits long and is configured to the SCE in four separate 4-byte arguments represented in HEX.
- Setting the salt is done through CLI. using a hexadecimal notation. It can be updated periodically. (By default it set to 0x12345678 0x12345678 0x12345678 0x12345678).
- The hash result is 128 bits long, and is represented in RDRs as a 32-byte string. Each byte represents four bits of the result. In order to make it printable, we add 64. (E.g. 0000 mapped to ASCII(64) = '@'; 1111 mapped to ASCII(127) = '_').
- The short-hash that appears in RDRs is composed of the lower 32 bits of the hash result.

6 Obtaining Documentation and Submitting a Service Request

For information on obtaining documentation, submitting a service request, and gathering additional information, see the monthly What's New in Cisco Product Documentation, which also lists all new and revised Cisco technical documentation, at:

http://www.cisco.com/en/US/docs/general/whatsnew/whatsnew.html

Subscribe to the What's New in Cisco Product Documentation as a Really Simple Syndication (RSS) feed and set content to be delivered directly to your desktop using a reader application. The RSS feeds are a free service and Cisco currently supports RSS Version 2.0.

Any Internet Protocol (IP) addresses used in this document are not intended to be actual addresses. Any examples, command display output, and figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses in illustrative content is unintentional and coincidental.

Americas Headquarters
Cisco Systems, Inc.
San Jose, CA

Asia Pacific Headquarters
Cisco Systems (USA) Pte. Ltd.
Singapore

Europe Headquarters
Cisco Systems International BV
Amsterdam, The Netherlands

Cisco has more than 200 offices worldwide. Addresses, phone numbers, and fax numbers are listed on the Cisco Website at **www.cisco.com/go/offices**.