Integrating Oracle Infinity Streams and Oracle Responsys

Implementation Guide

v1.02242019

Contents

Document Overview

Introduction

Oracle is pleased to present this self-service implementation guide to the Oracle Infinity and Oracle Responsys integration.

This document describes a step-by-step implementation of the Responsys retargeting solution using Infinity Streams. Please follow the steps outlined in the Tasks Required section in order to allow you to implement a Responsys product browse abandonment retargeting program using Infinity Streams and Data Connector to send data to Responsys.

Prerequisite:
This document assumes that your web developers have followed and implemented the Oracle Infinity Tag deployment strategy for collecting your website data. Please go to Oracle Infinity Tag Implementation Guide if you have not implemented the Oracle Infinity Tag in your website.

Additional Resources:

- Oracle Infinity Streams API v3.0 Implementation Guide
- Oracle Responsys Help Center

Oracle Infinity – Streams and Oracle Responsys Implementation Guide

Set up Infinity Streams and Responsys integration

Oracle Infinity is a data analytics platform for capturing, processing, storing and interrogating unlimited datasets for companies that want actionable customer intelligence to fuel their marketing programs. Infinity collects and reports on event data for websites, mobile apps, and other digital properties. The Infinity Streams provides a stream of data that you can access as each event happens and then further process as needed.

Oracle Responsys is an award-winning email marketing tools and functionalities that help improve conversions, strengthen customer relationships, and dramatically lessen messaging fragmentation.

Infinity Streams in combination with Responsys provides email remarketing solution to help marketers find, segment, and target customers at the right time based on their behavior on the website and mobile apps. You can leverage the Oracle Infinity Streams and Responsys integration in your marketing cloud solution to:

- Retarget and remarket to product browse and cart abandonment users.
- Reconnect with lapsed customers and customers who are not engaging through email.
- Remarket to customers on any part of the customer acquisition funnel.
- Post sale targeting, upgrade, refer a friend, next time discount, associated products.

Pre-requisite: **Tasks required to enable Responsys and Infinity Streams**

To enable Responsys and Infinity Streams you will need to perform each of the following steps detailed in the upcoming sections:

- STEP 1: Set up Responsys and Infinity connections
- STEP 2: Configure Responsys Product, Program, & Campaign settings
- STEP 3: Set up Streams Lab and Actions, Connections and Destinations

Each section will provide step-by-step instructions on how to properly configure Infinity Streams and Responsys integration.

STEP 1: Set up Responsys and Infinity connections

STEP 1.1: Create Infinity User Account in Responsys

To begin receiving retargeting data in your Responsys account from Infinity Streams, you'll first need to establish a connection between the two systems. This is done by creating a Responsys user account with web services access. The account will be used as the Login when creating a connection in Infinity Streams.

STEP 1.1.1: Creating Infinity User Account

1. Sign into your Responsys account.
2. Go to the Account Management page.
3. Under the User Management section, click on the Add User link.
4. For Login name, create one using the following naming scheme: infinity_api_<business name>
    a. Replace <business name> with the name of your business, no spaces or special characters.
5. For Email address, enter the email address of the user who will be managing this user account.
6. For User Status, select Active.
7. For User locale, select the desired language.
8. Under the Roles Assignment→Roles section, check Account Administrator.
9. Scroll to the top and click the **Create** button when complete. You should be returned to the Account Management page.

For further details on how to create a user account:
https://interact5.responsys.net/interact/help/rifh/en/Content/UserNew.htm

STEP 1.1.2: Create a Public/Private Key pair.

1. On a Windows machine use Puttygen and on a Mac or Linux machine use opensll from the command line to create a RSA 2048 key pair as shown below.

```
> openssl genrsa -out yourcompany_priv.pem 2048

> openssl req -new -x509 -key yourcompany_priv.pem -out yourcompany_pub.pem -days 1825
```

STEP 1.1.3: Managing User Authentication for the Infinity Streams User Account

1. From the Account Management page, under the User Management section, click the Manage User Authentication link.
2. Select the user account you just created and click the **Edit** button.
3. For User authentication, select Password or Certificate.
4. For User Certificate, select Upload user certificate and click the **Choose File** button.
5. Select the public key from your computer to upload.
6. Click the **Change** button when complete.

For further details on how to manage user authentication:
https://interact5.responsys.net/interact/help/rifh/en/Content/AuthUserEdit.htm

STEP 1.2: Set up Infinity connection to Responsys to begin passing data

To begin passing retargeting data from your Infinity Streams account to your Responsys account, you'll need to first establish a connection between the two systems. You begin by creating a Responsys user account with web services access as previously shown. Once that is done, you need to create a Connection in your Infinity account.

STEP 1.2.1: Creating a Connection to Responsys Account

1. Sign into your Infinity account.
2. Click on the **Action Center** button.
3. Near the right-hand corner, click on the Manage Connections link.
4. Click on the **New Connection** button.
5. Select Responsys from the drop-down menu.
6. For Name, it should default to "Responsys". Feel free to change.
7. For Login, enter the Login name that you used when you created the Infinity User Account in Responsys. i.e. infinity_api_<business name>
8. For Endpoint, select the API endpoint that is associated with the Pod for your Responsys account resides.
    a. Example: If your Responsys account is on Interact2, then your endpoint would be ws2.responsys.net.
9. For Certificate, select the private key earlier in the document onto the Certificate window.
10. Click the **Connect** button when complete.

STEP 1.3: Set up Responsys Custom Event Types

When Infinity Streams identifies a visitor for a retargeting opportunity, it will send it to Responsys via a direct API call using the Custom Event method. The Custom Event will also act as the trigger for sending the email.

Below is a list of Custom Event Types for common retargeting scenarios we recommend you create. Additional Custom Event Types can be created as you see fit. Using the "STREAMS_" prefix in the Custom Event name will allow you to better keep track of Custom Events that are used in Infinity Streams.

Custom Event Name	Description
STREAMS_Browse_Abandon	Browse abandonment
STREAMS_Cart_Abandon	Cart abandonment
STREAMS_Category_Abandon	Cart abandonment
STREAMS_Form_Abandon	Form abandonment
STREAMS_Ignore	For use in testing
STREAMS_Purchase	Purchase
STREAMS_Search	Search retargeting
STREAMS_Wishlist	Wishlist retargeting

STEP 1.3.1: Creating Custom Event Type

1. Sign into your Responsys account.
2. Go to the Account Management page.
3. Under the Account Customization→Global Settings section, click on the Define custom event types link.
4. Scroll down to the bottom and click on the Add new type link.
5. Add a new Custom Event by inputting a Custom Event Name and description.
6. Scroll up to the top and click the **Save** button to save the newly-created Custom Event.
7. Create additional Custom Event Types by repeating steps 4-6.

For further details on how to create a custom event type:

https://interact5.responsys.net/interact/help/rifh/en/Content/CustomEvent_Define.htm?Highlight=custom%20event%20type

STEP 1.4: Set up Destinations in Infinity to raise Responsys custom events

A Destination is used to set-up connection details such as which Responsys Custom Event to raise. An action can deliver data to only one destination. In this case, one destination should be created for each retargeting action.

STEP 1.4.1: Creating a Destination

1. In your Infinity account, click on the **Action Center** button.
2. Near the top, right-hand corner, click on the **Manage Connections** link.
3. Expand the Responsys section.
4. Click the **New** button next to the Destinations title.
    a. For Name, enter the desired name.
        i. Example: If this destination will be used for browse abandonment opportunities, you can name it "Browse Abandonment".
    b. For Folder Name:
        i. If the Format will be a Campaign Message, enter the name of the folder in which the campaign resides in your Responsys account.
        ii. If the Format will be a Custom Event, enter the name of the folder in which the Responsys profile list resides in your Responsys account.
        iii. If the Format will be a Table, enter the name of the folder in which the table resides in your Responsys account.
    c. For Format:
        i. Select Campaign Message if you want to trigger a campaign.
            1. For List Name, enter the name of the Responsys profile list in your Responsys account.
            2. For Campaign Name, enter the name of your campaign.
        ii. Select Custom Event if you want to raise a custom event.
            1. For List Name, enter the name of the Responsys profile list in your Responsys account.
            2. For Custom Event Name, enter one of the Responsys Custom Event Names you previously created.
                a. Example: If this destination will be used for browse abandonment opportunities, you would use STREAMS_Browse_Abandon.
            3. For the Event Key Value, enter the Responsys user ID you are going to use to key to map the website visitor:
                a. RIID_
                b. CUSTOMER_ID_
                c. EMAIL_ADDRESS_
                d. MOBILE_NUMBER_
            4. For Responsys Key, select the option that aligns with the Event Key Value you are using from the drop-down menu.
        iii. Select Table if you want to write data to a table.
            1. For Table Name, enter the name of the Responsys table in your Responsys account that you want to write to.
            2. For the Event Key Value, enter the primary key that has been created for the table.
    d. Click the **Save** button when complete.

5. Repeat steps 1-4 for other retargeting opportunities.

STEP 2: Configure Responsys Product, Program, & Campaign settings

Set up Responsys External Tracking Parameter

An external tracking parameter is a parameter that is appended to the URL string(s) of the link URLs that appear in your email messages. The parameter can pass values that can be used by 3rd-party solutions, such as web analytics solutions for tracking purposes.

Infinity Streams can take advantage of an external tracking parameter to improve the pool of known visitors by passing the identity of the visitor. The identity can come in the form of RIID_, CUSTOMER_ID_, MOBILE_NUMBER_, and EMAIL_ADDRESS_.

"infy" is the external tracking parameter we recommend you use.

STEP 2.1: Creating "infy" External Tracking Parameter

1. Sign into your Responsys account.
2. Go to the Account Management page.
3. Under the Account Customization→Campaign Management section, click on the Set external tracking parameters link.
4. Scroll down to the Available External Tracking Parameters for Classic Campaign section (if available).
    a. Click the Add Parameter link.
    b. For Name, enter "infy".
    c. Check the Required and Specify a restricted set of values checkboxes.
    d. For the value, use a Built-in Function to reference the identify you want to use.
        i. Example: For CUSTOMER_ID_, use $CUSTOMER_ID_$.
    e. Click the **Save** button when complete.
5. Scroll down to the Available External Tracking Parameters for New Campaign section (if available).
    a. Click the Add Parameter link.
    b. For Name, enter "infy".
    c. Check the Required and Specify a restricted set of values checkboxes.
    d. For the value, use Built-in Functions to reference the identify you want to use.
        i. Example: For CUSTOMER_ID_, use ${CUSTOMER_ID_}.
    e. Click the **Save** button when complete.

For further details to create an external tracking parameter:
https://interact5.responsys.net/interact/help/rifh/en/Content/SubAccountExternalTracking.htm?Highlight=external%20tracking%20parameter

STEP 2.2: Set up Responsys supplemental Product Details Table

When capturing activity related to an interaction of a product on your website, such as browsing and product details page or adding a product to a cart, we recommend simply capturing the product ID or SKU instead of all product detail data. The actual product detail data should be stored in a Responsys supplemental table that can be referenced by campaigns for personalization purposes using the product ID or SKU as the match key. The client is responsible for maintaining and updating this table of a regular basis.

In the example PRODUCT_DETAILS data schema presented below, SKU is set as the table's primary key*. The fields

featured are only suggested and can be populated as you see fit. Additional fields can also be added to accommodate the needs of your business.

Supplemental data table name: PRODUCT_DETAILS

Field Name	Field Type	Description
SKU*	Text Field (to 100 chars)	Product ID
PRODUCT_NAME	Text Field (to 500 chars)	Product name
PRODUCT_BRAND	Text Field (to 500 chars)	Product brand
PRODUCT_DESCRIPTION	Text Field (to 4000 chars*)	Product description
PRODUCT_AVAILABILITY	Text Field (to 100 chars)	Product availability
PRODUCT_PRICE	Number Field	Current price, sales price
PRODUCT_URL	Text Field (to 500 chars)	Product details page URL
PRODUCT_IMG_URL	Text Field (to 500 chars)	Product image URL

STEP 2.2.1: Creating Product Details Table to Campaign

(Note: Optional step if you already have product details table)

1. Sign into your Responsys account.
2. From the left-hand toolbar, choose Data→Create view/Supplemental Data.
3. Under the Create Supplemental Table→Empty Table section, click the Specify Fields link.
4. Enter the desired Field Names and assign the appropriate Data Types. Feel free to use the example PRODUCT_DETAILS table above for reference.
5. Click the **Next** button when complete.
6. Assign the appropriate field(s) as the Primary Key Field(s). If you used the example PRODUCT_DETAILS table, you would assign SKU as the Primary Key Field.
7. For "Select the folder to hold the new data source, select the desired folder." We recommend the !MasterData folder.
8. For "Enter a name for the new data source", enter PRODUCT_DETAILS.
9. For "Choose an expiration date for the new data source", check Never expires.
10. Click the **Create** button when complete.

For further details on how to create a supplemental table:
https://interact5.responsys.net/interact/help/rifh/en/Content/FileNewMethodSelect.htm?Highlight=create%20supplemental%20data%20table

STEP 2.3: Set up Responsys Campaigns to be sent when a retargeting event is triggered

The Retargeting Program can be designed to trigger the sending of campaigns. For some of these campaigns, you may want to feature the product details of the SKUs associated to the retargeting opportunity. For example, you may want to include the product name, price, and URL of the SKUs that the user has abandoned. To do this, it requires two steps:

- Delegating Product Details Table to Campaign
- Integrating Product Details Using Personalization Codes

STEP 2.3.1: (required if using New Campaign) Delegating Product Details Table to Campaign

The way the PRODUCT_DETAILS table is referenced for personalization is different based on whether you're creating a campaign using 'Classic Campaign' or 'New Campaign'. If you are using the New Campaign, then you will need to delegate

the PRODUCT_DETAILS table to the campaign.

To use the PRODUCT_DETAILS table for personalization:

1. Create new or open existing campaign.
2. In the Data sources section, click on the **Edit** button.
3. Click on the **Add new source** button.
    a. For type of source, select Supplemental data option.
    b. Select the desired supplemental data table. This would be the PRODUCT_DETAILS table if you followed that example.
    c. Check the 'Data source is used only as a lookup table' option.
    d. Click the **Select** button.
4. Under the Supplemental tables section, locate the supplemental table you just delegated.
5. For 'Data source alias', enter PRODUCT_DETAILS.
6. Click on the **Add columns** button and select all the columns in the table that will be used for personalization.
    a. Click the **Select** button when complete.
7. Click the Lookup key checkbox next to the column that will be used as the lookup key. If you used the example PRODUCT_DETAILS table, it should look something like this.

    Data source alias: PRODUCT_DETAILS

Column	Alias	Lookup Key
SKU	SKU*	*
PRODUCT_NAME	PRODUCT_NAME	
PRODUCT_BRAND	PRODUCT_BRAND	
PRODUCT_DESCRIPTION	PRODUCT_DESCRIPTION	
PRODUCT_AVAILABILITY	PRODUCT_AVAILABILITY	
PRODUCT_PRICE	PRODUCT_PRICE	
PRODUCT_URL	PRODUCT_URL	
PRODUCT_IMG_URL	PRODUCT_IMG_URL	

8. Click the **Save** button when complete.

For further details on how to work with data sources:
https://interact5.responsys.net/interact/help/rifh/en/Content/EMD_Data_Srcs.htm

STEP 2.3.2: Integrating Product Details Using Personalization Codes

As previously mentioned, the way the PRODUCT_DETAILS table is referenced for personalization is different based on whether you're creating a campaign using 'Classic Campaign' or 'New Campaign'. For Classic Campaign, you would use built-in functions (BiFs) to retrieve the product details data fields from the table. For 'New Campaign', you would first delegate the PRODUCT_DETAILS table to the campaign, as previously shown, then use the Responsys Programming Language (RPL) to retrieve the product details data fields.

STEP 2.3.3: Personalization in Classic Campaign

Let's assume you have the campaign built, a HTML document assigned to the campaign, and all related materials are saved to the "Retargeting_Programs" folder. Let's also assume that a comma-delimited list of SKUs is being passed into the

Retargeting Program via a Program entry variable named “RETARGETING_SKU”.

1. Sign into your Responsys account.
2. Create a subdocument for the product details layout:
3. From the left-hand toolbar, choose Content→Manage Content.
    a. Within the Manage Content page, click on the **New Folder** button and create a new folder named "Retargeting_Programs".
    b. Within the refreshed folder listing in the Manage Content page, click on the "Retargeting_Programs" folder.
    c. Click on the **Create Document** button.
        i. For the Document name, enter “SKU_Details”.
        ii. For the Content, highlight and paste the code below and paste it in the content window, overwriting what currently exists:

Line	Code
1	<html>
2	<body>
3	$SETVARS(LOOKUP(Retargeting_SKU))$
4	$SETVARS(VARLIST(7, PRODUCT_NAME, PRODUCT_BRAND, PRODUCT_DESCRIPTION, PRODUCT_AVAILABILITY, PRODUCT_PRICE, PRODUCT_URL, PRODUCT_IMG_URL, LOOKUPRECORDS(!MasterData, PRODUCT_DETAILS, SKU, LOOKUP(RETARGETING_SKU), PRODUCT_NAME, PRODUCT_BRAND, PRODUCT_DESCRIPTION, PRODUCT_AVAILABILITY, PRODUCT_PRICE, PRODUCT_URL, PRODUCT_IMG_URL)))$
5	$PRODUCT_NAME$ $PRODUCT_BRAND$ $PRODUCT_DESCRIPTION$ $PRODUCT_AVAILABILITY$ $PRODUCT_PRICE$ $PRODUCT_URL$ $PRODUCT_IMG_URL$
6	</body>
7	</html>

Line	Field Reference	Description
4	$SETVARS(LOOKUP(Retargeting_SKU))$	Reference the individual product ID from the "Retargeting_SKU" loop.
4	$SETVARS(VARLIST(7, PRODUCT_NAME, PRODUCT_BRAND, PRODUCT_DESCRIPTION, PRODUCT_AVAILABILITY, PRODUCT_PRICE, PRODUCT_URL, PRODUCT_IMG_URL, LOOKUPRECORDS(!MasterData, PRODUCT_DETAILS, SKU, LOOKUP(RETARGETING_SKU), PRODUCT_NAME, PRODUCT_BRAND, PRODUCT_DESCRIPTION, PRODUCT_AVAILABILITY, PRODUCT_PRICE, PRODUCT_URL, PRODUCT_IMG_URL)))$	The individual SKU from the RETARGETING_SKU Program entry variable will be used to reference the PRODUCT_DETAILS data table in the "!MasterData" folder to pull the values from the PRODUCT_NAME, PRODUCT_BRAND, PRODUCT_DESCRIPTION, PRODUCT_AVAILABILITY, PRODUCT_PRICE, PRODUCT_URL, and PRODUCT_IMG_URL fields.

Line	Field Reference	Description
5	$PRODUCT_NAME$ $PRODUCT_BRAND$ $PRODUCT_DESCRIPTION$ $PRODUCT_AVAILABILITY$ $PRODUCT_PRICE$ $PRODUCT_URL$ $PRODUCT_IMG_URL$ 	This retrieves the values from the fields in the PRODUCT_DETAILS table. Feel free to adjust to fit desired layout.

d. Click on the **Save** button when complete.

4. Edit the HTML document of your existing campaign.
5. Locate the place in your code where you want the product details of the retargeting SKUs to appear.
6. Highlight and paste the code below and paste it to the desired location:

Code	Description
$FOREACHNOBR(Retargeting_SKU, PAIRSLIST(1, SKU, LOOKUP(RETARGETING_SKU)), /contentlibrary/Retargeting_Programs,SKU_Details)$	Create a loop named "Retargeting_SKU" that parses the comma-delimited list of SKUs from the RETARGETING_SKU Program entry variable and then passes the value to the "SKU_Details" subdocument found in the "/contentlibrary/Retargeting_Programs" folder.

7. Click the **Save** button when complete.

STEP 2.3.3.1: Personalization in New Campaign

Let's assume you have the campaign built, a HTML document assigned to the campaign, the PRODUCT_DETAILS table has been delegated to the campaign, and all related materials are saved to the "Retargeting_Programs" folder. Let's also assume that a comma-delimited list of SKUs is being passed into the Retargeting Program via a Program entry variable named "RETARGETING_SKU".

1. Sign into your Responsys account.
2. Open your existing campaign.
3. From the Campaign Workbook screen, click on the Edit link in the Message section.
4. Locate the place in your code where you want the product details of the retargeting SKUs to appear.
5. Highlight and paste the code below and paste it to desired location:

Line	Code
1	<!--BEGIN RETARGETING CODE-->
2	<#list RETARGETING_SKU?split(",") as RETARGETING_SKU_EACH>
3	<#data PRODUCT_DETAILS as PROD_DETAILS>
4	<#filter SKU=RETARGETING_SKU_EACH>
5	<#fields PRODUCT_NAME PRODUCT_BRAND PRODUCT_DESCRIPTION PRODUCT_AVAILABILITY PRODUCT_PRICE PRODUCT_URL PRODUCT_IMG_URL>

6	${PROD_DETAILS.PRODUCT_NAME} ${PROD_DETAILS.PRODUCT_BRAND} ${PROD_DETAILS.PRODUCT_DESCRIPTION} ${PROD_DETAILS.PRODUCT_AVAILABILITY} ${PROD_DETAILS.PRODUCT_PRICE} ${PROD_DETAILS.PRODUCT_URL} ${PROD_DETAILS.PRODUCT_IMG_URL}
7	</#data>
8	</#list>
9	<!--END RETARGETING CODE-->

Line	Field Reference	Description
2	<#list RETARGETING_SKU?split(“,”) as RETARGETING_SKU_EACH>	Reference the comma-delimited list of retargeting SKUs from the RETARGETING_SKU Program entry variable and parse the data, making the individual SKU available in a variable named RETARGETING_SKU_EACH.
3	<#data PRODUCT_DETAILS as PROD_DETAILS>	Reference the PRODUCT_DETAILS table and assign it the alias PROD_DETAILS.
4	<#filter SKU=RETARGETING_SKU_EACH>	Retrieve records from the PROD_DETAILS table where RETARGETING_SKU_EACH variable matches a value in the SKU field (in the table).
5	<#fields PRODUCT_NAME PRODUCT_BRAND PRODUCT_DESCRIPTION PRODUCT_AVAILABILITY PRODUCT_PRICE PRODUCT_URL PRODUCT_IMG_URL>	Retrieve the fields from the PROD_DETAILS table.
6	${PROD_DETAILS.PRODUCT_NAME} ${PROD_DETAILS.PRODUCT_BRAND} ${PROD_DETAILS.PRODUCT_DESCRIPTION} ${PROD_DETAILS.PRODUCT_AVAILABILITY} ${PROD_DETAILS.PRODUCT_PRICE} ${PROD_DETAILS.PRODUCT_URL} ${PROD_DETAILS.PRODUCT_IMG_URL} 	This retrieves the values from the fields in the PROD_DETAILS table. Feel free to adjust to fit desired layout.

6. Click the **Save** button when complete.

STEP 2.4: Set up a Responsys Retargeting Program to listen to custom events

As previously mentioned, when Infinity Streams identifies a visitor for a retargeting opportunity, it will send that to your Responsys account via a Custom Event. To listen for the Custom Event, you are required to set up a Program. From the Program, you can then trigger the sending of a retargeting email.

STEP 2.4.1: Setting up Retargeting Program

1. Open/Create your retargeting Program using Program Designer.
2. Click on the **Settings** button on the top right-hand corner.
    a. Expand the Tracking and variables section.
    b. Add the desired entry tracking variable(s) and assign the appropriate type(s).
        i. Note: The entry tracking variable name(s) should mirror the name(s) you mapped when creating the Infinity Action associated to the Custom Event type you are planning to raise.

    c. Continue configuring the Settings and click the **OK** button when complete.
        i. Note: Please remember to setup the "unplublish" settings as per your company standards.
3. From the left-hand toolbar in Program Designer, drag the Custom event stage onto the canvas.
4. Double-click on the Custom event stage to configure:
    a. For the Stage label, enter the desired label name.
        i. Example: If this stage will be used to listen for browse abandonment events, you may want to use the label name "Browse Abandonment".
    b. OPTIONAL: To add a description, enter the desired description.
    c. For 'Listen for custom event type', select the desired custom event types you created earlier.
        i. Example: If this stage will be used to listen for browse abandonment events, you should select "STREAMS_Browse_Abandon".
    d. For Audience label, enter the desired audience label.
        i. Example: If this stage will be used to listen for browse abandonment events, you may want to use the audience label "Browse Abandonment".
    e. For When a person is already in the program, check one of the following:
        i. 'Create a new entry in the program' if you want the Program to enter a user each and every time they are qualified for a retargeting opportunity even if the user is still in the Program.
        ii. 'Do not create a new entry in the program and leave the existing entry where it is if you don't want the Program to enter the same user if they are currently still in the Program.'
        iii. 'Do not create a new entry in the program and move the existing entry to this location' if you want the Program to move the user from where they currently are in the Program to this stage.
    f. Click the **Done** button when complete.
5. Continue designing your Retargeting Program as desired.

For further details on how to create a program and using a custom event:
https://interact5.responsys.net/interact/help/rifh/en/Content/EMD_Data_Srcs.htm

Things to Consider

There may be occasions where you would like to remove a user from a Retargeting Program if another retargeting opportunity has been identified. For example, let's say you have a Cart Abandonment Program that sends a series of 3 emails across a span of 5 days. If a user makes a purchase 2 days after entering the Program, you may want to remove them from the Program to prevent them being sent the remaining emails, which may cause confusion. You can do this by adding another Custom event stage to the Cart Abandonment Program. The stage should listen to the Purchase retargeting opportunities (i.e. STREAMS_Purchase Custom Event type) and 'Do not create a new entry' in the program and 'move the existing entry to this location' checked. The stage should connect directly to an End stage.

STEP 3: Set up Streams Lab and Actions, Connections and Destinations

The Streams Lab is where you first start to define your retargeting opportunity.

The Streams UI provides access to real-time online activity enriched with visitor information. It is not a database. Rather, it is a stream of data that you can access as each event happens and then further process as needed. You can segment and filter streaming data on any collected attribute.

Streams enables your marketing team to continually review and update products or promotions based on their performance in monitored campaigns. For example, you can look at geographic, device, and referrer data for a specific use case.

STEP 3.1: Creating a Streams Query

Let's assume the website has been tagged per the Oracle Infinity Tag Guide, data.wt.pn_sku is the parameter being using the capture the product SKU and data.infy is used to capture the visitor ID when the tag is fired. We will also be passing the list of SKUs as a comma-delimited list in a single parameter.

1. In your Infinity account, click on the **Streams** button.
2. On the top, left-hand corner, click on the **START** button. You will see data being captured in real time as the tag parameters appear on the left-hand column.
3. In the left-hand column, locate the following parameters and drag them onto the Streams canvas to the right:
    a. data.infy
    b. data.wt.pn_sku
    c. data.wt.tx_e

    Additional parameters can also be moved to the canvas as needed. Parameter values that need to be passed to Responsys must be included on the canvas.

4. In the Select bar near the top, enter the desired query to define your retargeting opportunity. Refer to the Streams Query Format section in the APPENDIX for more details.

    Example
    The Infinity tags have been integrated into a website. The tag utilizes the data.infy parameter to capture the visitor identity, the data.wt.pn_sku parameter to capture the product SKU, and the data.wt.tx_e parameter to keep track of the activity type. The following are the different possible values for the data.wt.tx_e parameter:

Value	Definition
v	Viewing a product details page.
a	Adding a product to the cart
p	Purchase of a product

    Let's say you want to create a Streams query to identify visitors who qualify for a Browse Abandonment opportunity. A Browse Abandonment opportunity is defined by known visitors who viewed a product details page, but did not add any product to the cart at all during the session, which is now closed. The query results should include the visitor identity and the product SKU that was viewed. Each product SKU that a visitor has viewed will appear as a distinct data record. Here's an example of the query:

Query
data.infy, data.wt.pn_sku WHERE session.closed='true' AND EXISTS(data.infy) AND ANY(data.wt.tx_e='v') AND ALL(data.wt.tx_e!='a')

If you would like to aggregate all the product SKUs a visitor has viewed during the same session as a comma-delimited list in a single record, here's an example of the query:

Query
data.infy, CONCAT_DISTINCT(data.wt.pn_sku, ',') AS 'SKU' WHERE session.closed='true' AND EXISTS(data.infy) AND ANY(data.wt.tx_e='v') AND ALL(data.wt.tx_e!='a') HAVING data.wt.tx_e='v'

5. Click the **Save** button when complete.
    a. Name your Streams Label and click the **Save** button.

STEP 3.2: Set up an Action in Infinity

An Action is used to automate a Streams query, connect it to the pre-configured Responsys destination and map the fields selected in the Streams query to the Responsys Program Entry Variables.

STEP 3.2.1: Creating an Action

1. In your Infinity account, click on the **Action Center** button.
2. Click on the **Create Action** button.
    a. Under the (1) Name and Description section:
        i. For Name, enter a name for your Action.
    b. Under the (2) Destination section:
        i. For Send data to, select the pre-configured Responsys destination and sub-connection you want to send the data to.
    c. Under the (3) Data section:
        i. For Data to send, select the desired Streams Lab.
        ii. Map the parameters selected in your Streams Lab to the appropriate Responsys Program Entry Variables.
            1. Note: data.infy should map to Event Key Value that you used when creating the Responsys sub-connection.
    d. Under the (4) Save section:
        i. Click the **Save** button when complete.
3. When you return to the Action Center main page, you can activate the Action you created by clicking the **START** button (next to the Action). This will begin sending data to Responsys when the Streams query conditions are met.

APPENDIX 1: Oracle Infinity Tag Implementation Guide

Set up Oracle Infinity Tag data collection

Oracle Infinity Tag collects and reports on event data for websites, mobile apps, and other digital properties. When a user performs an action such as a button click or a page view, a request containing a string of parameters and values is created and transmitted to the data collection server. These parameters can provide simple information about sessions and page views or customized information such as commerce transactions, media activity, and device hardware. Infinity can report on this data in real time or trend it over days, weeks, months, and years.

Tasks required to configure the Oracle Infinity Tag

To setup the Infinity JavaScript Tag on your website you will need to perform each of the following steps detailed in the upcoming sections:

- STEP 1: Tagging your web page
- STEP 2: Deploying the Oracle Infinity Tag
- STEP 3: Verifying the Oracle Infinity Tag

Each section will provide step-by-step instructions on how to properly setup the Oracle Infinity JavaScript Tag.

STEP 1: Tagging your web page with the Oracle Infinity Tag

1. Contact your Oracle Infinity solutions consultant to obtain your Oracle Infinity Tag.
2. Add the tag URL to a script block similar to the following example:

```
<script type="text/javascript" async
src="//c.oracleinfinity.io/acs/account/{account_GUID}/{tag_ID}/odc.js"></script>
```

3. Paste your Oracle Infinity Tag after the opening <head> tag of your web page code to ensure the tag loads and fires before the visitor moves off the page. It can be placed anywhere your site coding guidelines allow, but you should avoid locations that could interfere with the correct order of execution.

STEP 2: Deploying the Oracle Infinity Tag

The Oracle Infinity Tag collects data from applications capable of executing JavaScript. You can use it to track site content and deliver data to Oracle Infinity Analytics for use with reports. You can use the collected data to drive marketing activities and integrate with other Oracle Marketing Cloud applications.

The Oracle Infinity Tag:

- Uses a single line of JavaScript that should not need to be updated or changed.
- Uses a content delivery network (CDN) to dynamically deliver tags, which means that you can make changes without retagging your sites.
- Is associated with an account ID and a tag ID, which are unique strings for your site and tag.
- Will set Oracle-specific first-party cookies.
- Has a tag context that provides a unique tag configuration selectable by query parameter.
- Supports optional tag plugins to support additional functions.
- Supports secure cookies by default. To disable secure cookies in order to host content on non-HTTPS sites, contact your Oracle Infinity solutions consultant.

To obtain an Oracle Infinity Tag for your site:

1. Contact your Oracle Infinity solutions consultant and send them your:
   - Oracle Infinity account name
   - An email address where the tag should be delivered
   - The domain of the pages you need to track
   - (Optional) A list of tag plugins you need to have enabled

   Note: Each added plugin increases the tag size and load time, so you should only enable plugins you really need.

A tag URL will be returned to you that will look something like the following sample:

c.oracleinfinity.io/acs/account/{account_GUID}/{tag_ID}/odc.js

Where:

- An account GUID identifies your account. All your tags will use the same account GUID so that all data collected for your account is stored together.

- A tag ID allows you to assign access rights to your Oracle Infinity Tags and put them into a hierarchical format.
- odc.js contains tag configuration data, context information, and functions related to loading your tag. Context refers to a unique Oracle Infinity Tag configuration that is selectable according to query parameter. You can configure multiple contexts, but the Oracle Infinity Tag supports only one active context at a time.

Important: JavaScript object names are case-sensitive.

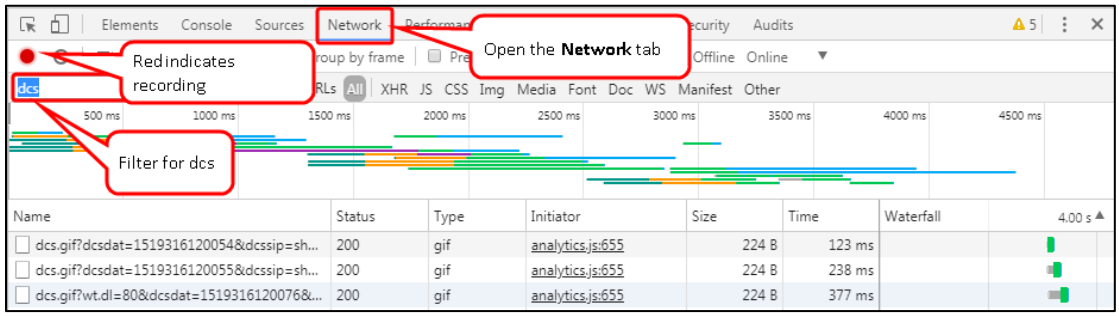
STEP 3: Verifying the Oracle Infinity Tag

You can use a browser to determine whether your Oracle Infinity Tag is correctly deployed on your site. Use an incognito window in Chrome and use its built-in developer tools.

Important: Disable any ad blockers or privacy plugins you may have installed in Chrome, because they can prevent the tag from firing.

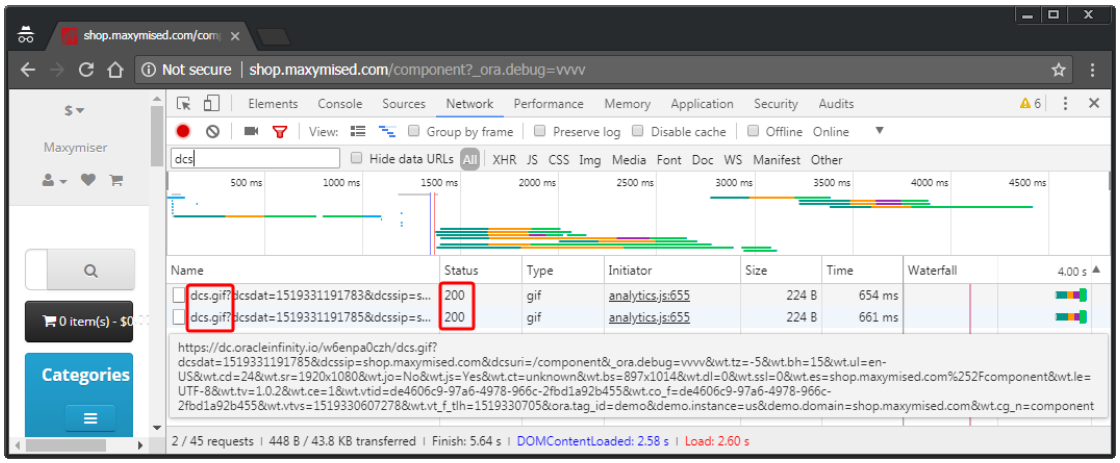
When the Oracle Infinity Tag loads on a page, it begins tracking page load and activity events data using preconfigured methods.

1. In Chrome, open an incognito window and navigate to your web page.
2. Press F12 to open the debug console.
3. Select the Network tab.



A red circle indicates recording.

4. Make sure that recording is enabled and then press F5 to refresh the tagged page. Events are displayed as the page loads.
5. Enter dcs in the filter box to locate the Oracle Infinity collection events. Instances of dcs.gif with a status of 200 indicate successful page load tracking. Depending on your tag configuration, you may see additional events being tracked as you browse your site, such as scroll events, right-click actions, drop-down menu tracking, mouse over actions, and other user interactions.

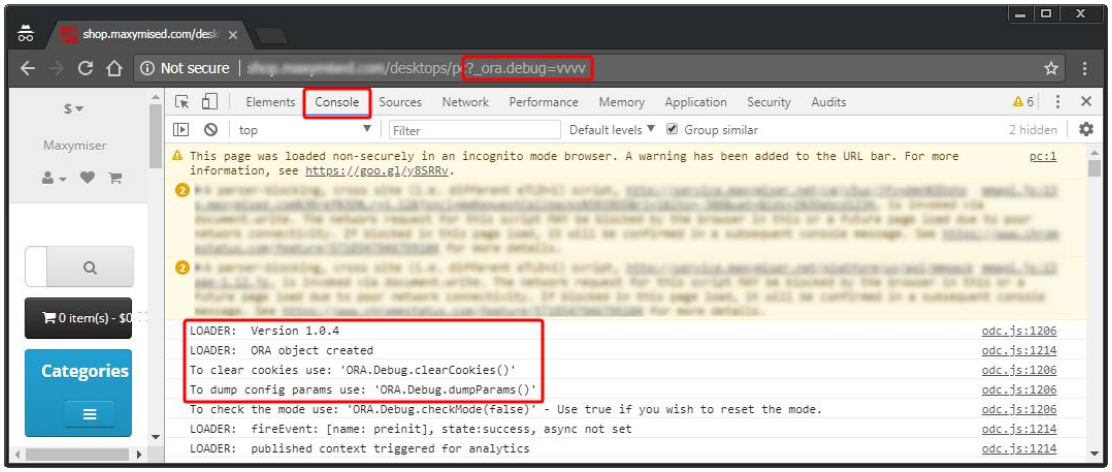


STEP 3.1: Verifying and debugging the Oracle Infinity Tag

You can append the ?_ora.debug=vvvv query parameter to the URL of your web page to verify the Oracle Infinity Tag and provide verbose debugging information.

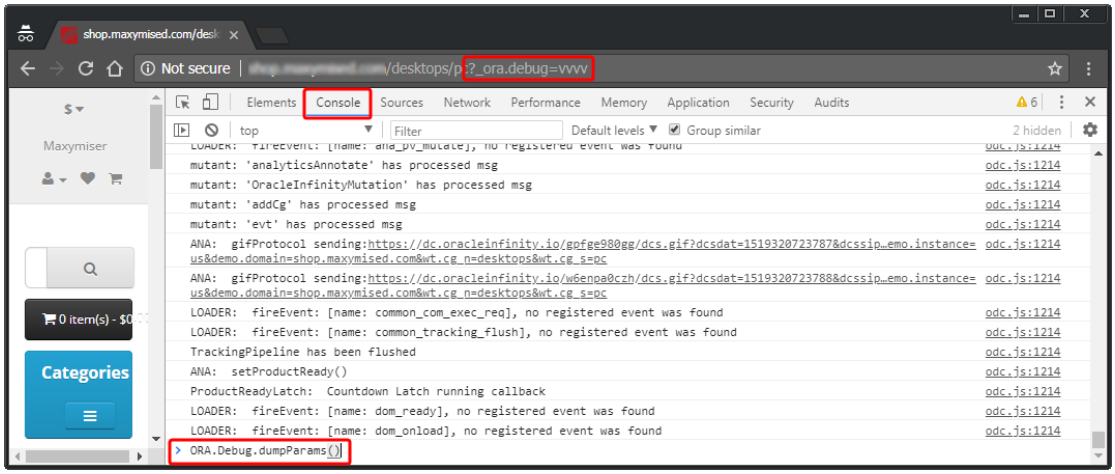
To verify the functionality of your tag and view debugging information:

1. In Chrome, open an incognito window and navigate to your web page.
2. Press F12 to open Chrome's developer tools.
3. Append ?_ora.debug=vvvv as a query parameter to the URL, where the number of Vs represents the debug output level displayed within the browser's Console tab. The more Vs, the more granular the output (up to a maximum of four Vs).

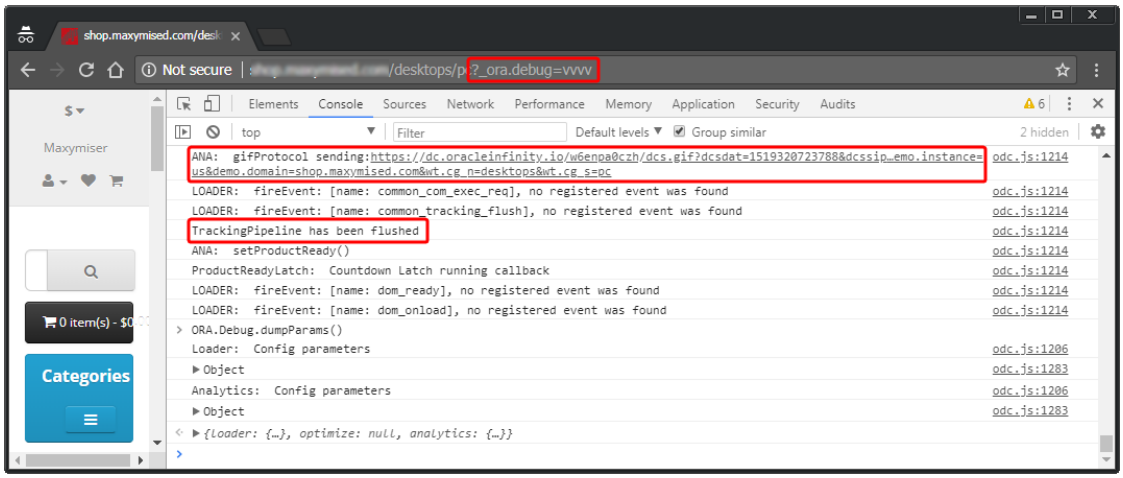


The debug output and console logs show the hosted tag version, such as LOADER: Version 1.0.4.

4. At the bottom of the debug log, type ORA.Debug.dumpParams() after the Image of the debug console's prompt for text entry and then press Enter to output the configuration parameters for the Oracle Infinity Tag.



5. Search the debug output for the following lines to ensure the tag loads successfully:
   - ANA: gifProtocol sending:http://URI/dcs.gif?query_args
   - TrackingPipeline has been flushed



The lack of an error typically indicates a successful load and track if these lines are not followed by or preceded by a product timeout message.

Data collected by default

By default, the following parameters are collected by the Oracle Infinity Tag (automatic events). This list does not include data derived after this default data set is collected.

Parameters collected by default

Parameter	Description
dcssip	The domain visited, such as www.oracle.com.
dcsuri	The portion of the URL after the domain, URI stem, such as /folder/page.html.
wt.bh	wt.bh indicates the web client's browsing hour (local time of day on a 24-hour clock).
wt.cd	wt.cd indicates the web client's screen color depth.
wt.ce	wt.ce indicates whether the visitor has first-party cookies enabled (1) or disabled (0).
wt.co_f	wt.co_f is a random value used as a visitor ID and stored in a first-party cookie.
wt.ct	wt.ct specifies the connection type used to send data.
wt.dl	wt.dl specifies a numeric identifier for the kind of event tracked, which are used for event-level filtering and reporting.
wt.es	wt.es identifies the page a user was on when an event occurred.
wt.jo	wt.jo indicates whether the web client has enabled Java (yes) or not (no).
wt.js	wt.js indicates whether the web client supports or has enabled JavaScript (yes) or not (no).
wt.p_did	wt.p_did provides a hash value that represents the device used by the visitor. Its values are only collected if the persona service is enabled.
wt.p_pid	wt.p_pid provides a hashed persona identifier generated by the persona service. wt.p_pid is typically identical to wt.p_vid Its values are only collected if the persona service is enabled.
wt.p_sid	wt.p_sid provides a unique ID for the session for use by the persona service. Its values are only collected if the persona service is enabled.
wt.p_vid	wt.p_vid is generated if the wt.p_did does not match an existing persona. If the persona is matched, wt.p_vidis identical to wt.p_pid. Its values are only collected if the persona service is enabled.
wt.sr	wt.sr indicates the web client's screen resolution.
wt.ssl	wt.ssl indicates whether the hit's content is secured via HTTPS (1) or not (0).
wt.ti	wt.ti identifies the HTML page title of the associated web content or the name of the mobile app in the case of the SDK.
wt.tv	wt.tv specifies the version of the JavaScript tag that is currently deployed.
wt.tz	wt.tz indicates the web client's time zone offset from UTC.
wt.ul	wt.ul indicates the app or web client's user language.

Standard Parameters

Oracle Infinity Analytics has the following categories of preconfigured report parameters for use in dimensions, measures, and segments:

- Commerce
- Content
- Device
- Geo
- Mobile App
- Time
- Traffic Source
- Users

Parameter prefixes

Oracle Infinity Analytics' standard parameters include the following prefixes that indicate data processing and augmentation has occurred since the data was collected:

- data.: Collected by the Oracle Infinity Tag or an SDK
- ext.: Device-related parameters added based on user agent lookup and geo parameters added based on IP address lookup

Example: The Oracle Infinity Tag or an SDK collects values for the wt.cg_n parameter, processes them, and adds the data. prefix to the parameter. The parameter is displayed in the Oracle Infinity Analytics UI as data.wt.cg_n.

Preconfigured Standard Parameters for each Category

Commerce category

Parameter name	Parameter	Advanced options
Campaign ID	data.wt.mc_id	
Conversion	data.wt.conv	
Invoice	data.wt.tx_i	Split by semicolon
Product SKU	data.wt.pn_sku	Split by semicolon
Scenario Name	data.wt.si_n	Split by semicolon
Scenario Step Name	data.wt.si_p	Split by semicolon
Scenario Step Number	data.wt.si_x	Data type is numerical Split by semicolon
Transaction Event	data.wt.tx_e	

Transaction Subtotal	data.wt.tx_s	Data type is numerical Split by semicolon
Units	data.wt.tx_u	Data type is numerical Split by semicolon

Content category

Parameter name	Parameter	Advanced options
Content Group	data.wt.cg_n	Split by semicolon
Content Sub Group	data.wt.cg_s	Split by semicolon
Domain	data.domain	
Onsite Search Phrase	data.wt.oss	
Onsite Search Results	data.wt.oss_r	Data type is numerical
Page Title	data.wt.ti	
Page URI	data.page-uri	
Page URL	data.wt.es	

Device category

Parameter name	Parameter
Browser Name	ext.browser.name
Browser Version	ext.browser.version
Client IP	data.client-ip
Device Name	ext.device.mktn
Device Type	ext.device.type
Network Type	ext.net.type
Operating System Name	ext.os.name
Operating System Version	ext.device.osver
Organization	ext.net.org

Geo category

Parameter name	Parameter
City	ext.geo.city
Country	ext.geo.country
Language	data.wt.ul
Region	ext.geo.region

Mobile App category

Parameter name	Parameter
Mobile App Name	data.wt.a_nm
Mobile App Version	data.wt.av
Mobile Carrier	data.wt.a_dc
Mobile App Event Type	data.wt.sys
Mobile SDK Build	data.wt.sdk_v

Time category

Parameter name	Parameter
Date	datetime_utc

Traffic Source category

Parameter name	Parameter
Referrer	data.referrer
Referring Domain	data.referrer-domain
Referring URI	data.referrer-path
Search Engine	ext.search.engine
Search Phrase	ext.search.phrase
Traffic Source Type	ext.source.type

Users category

Parameter name	Parameter
New-Returning User	user_type
Visitor ID	session.visitor_id

APPENDIX 2: Streams API v3.0 Implementation Guide

Use the Streams API to integrate streams with your applications.

Overview

Schema documentation: Streams Schema v3.0

NOTE: Changes from Schema 2.0:

- The "Event" Block is removed, so "data", "ext", and the "top level event parameters" are all on the same level as the session block now.
- epoch_datetime_utc has been added as a top level parameter

Parameter documentation: Data Pipeline Parameters and Sources

Streams lab and session streams support the following streams API query statements, clauses, and operators:

- `select`: Specifies the part of the stream to return. For example, content group (`wt.cg_n`) and sub-content group (`wt.cg_s`).
- `*`: Specifies a wildcard search in `select`. For example, to return all parameters that start with `data.`, use `select data.*`.

  Important: The wildcard must follow a period (`.`)and cannot be followed any additional characters as shown in the following examples:

  - **Correct**: `select data.*` and `select data.wt.*`
  - **Incorrect**:`select data.*.x` and `select data*`

- `where`: Specifies how to filter the stream (for example, include only events from the United Kingdom with a certain campaign ID).
- `=`: Is equal to
- `!=`: Is not equal to
- `like`:Is like (use with `*` wildcard)
- `notlike`: Is not like (use with `*` wildcard)
- `EXISTS()`: This Boolean function checks for the existence of a parameter. For example, if you want to see a stream of events only where the user is in the shopping funnel:
  `select * where EXISTS(data.wt.tx_e)`

- `NOT_EXISTS()`: This Boolean function checks for the absence of a parameter. For example, if you want to identify pages that do not have a title and fix them:
  `select * where NOT_EXISTS(data.wt.ti)`

Tip: A good query selects only the necessary data to be streamed.

Session streams only

Session streams supports the `ANY` and `ALL` operators:

- `ANY`:

  Examples:

  - View only sessions that have at least one product view: `select * where ANY(data.wt.tx_e=v)`
  - You want to view only sessions that have at least one product view event and the content group for that particular view is Shoes: `select * where ANY(data.wt.tx_e=v and data.wt.cg_n=Shoes)`
- `ALL`:

  Example: View only sessions that have no product view event: `select * where ALL(data.wt.tx_e!='v')`

Streams Query Format

Streams queries are composed in the following format:

SELECT <select criteria> ***WHERE*** <where criteria> ***HAVING*** <having criteria>

Select Criteria – Required

The select criteria are a comma separated list of values to be included in the message.

Type of Select criteria	Example	Behavior	Selection applied to	Notes
Basic segment	data.wt.ti	The parameter is included in the outgoing message if present in the current event	Current event if matching the WHERE criteria	
Basic prefix wildcard	data.*	All parameters in the current event that match the filter are included in the outgoing message	Current event if matching the WHERE criteria	
All wild card	*	All parameters in the current event are included in the outgoing message	Current event if matching the WHERE criteria	
Any selector	ANY(data.wt.ti)	Include a single event value from the session. Which value specifically is returned is irrelevant	Current event, or any other event in the session with the parameter, if matching the WHERE criteria.	Wild cards are not supported
Earliest selector	EARLIEST(data.wt.ti)	Include the value from the earliest event in the session that has the parameter	All events matching the HAVING criteria	Wild cards are not supported

Earliest with null	EARLIEST_INCLUDE_NULL(data.wt.ti)	Include the value from the first event in the session evaluated if present	All events matching the HAVING criteria	Wild cards are not supported
Count Selector	COUNT(data.wt.ti)	Include the number of events in the *session* that have a value for the specified key	All events matching the HAVING criteria	Wild cards are not supported
Count Distinct	COUNT_DISTINCT(data.wt.ti)	Include the number of unique values for the specified key	All events matching the HAVING criteria	Wild cards are not supported Case insensitive matching
Count all	EVENT_COUNT()	Include the number of events evaluated	All events matching the HAVING criteria	Wild cards are not supported
Latest selector	LATEST(data.wt.ti)	Include the last value found	All events matching the HAVING criteria	Wild cards are not supported
Latest with null	LATEST_INCLUDE_NULL(data.wt.ti)	Include the value from the last event evaluated if present	All events matching the HAVING criteria	Wild cards are not supported
Concat	CONCAT(data.wt.ti, ';')	Include all values for the specified key concatenated into a single string with the provided separator	All events matching the HAVING criteria	Wild cards are not supported Separators can be strings of multiple characters Order and case are maintained

Concat distinct	CONCAT_DISTINCT(data.wt.ti, ',')	Include all distinct values for the specified key concatenated into a single string with the provided separator	All events matching the HAVING criteria	Wild cards are not supported Separators can be strings of multiple characters Order and case are maintained Case is not used when determining uniqueness. If the value for one event is 'AaA' and the value of the next event is 'aAa', we will only include 'AaA' in the distinct list of values

Aliasing - Optional

Select criteria can take an optional alias with the format: <selection> AS '<alias>'.

<selection> can be an from the table above, without using wildcards, and <alias> can be any string

For example:

SELECT session.visitor_id AS 'visitor.id', session.session_id as 'visitor.session', EARLIEST(data.wt.cg_n) AS 'content.group.first', LATEST(data.wt.cg_n) AS 'content.group.last'

yields this message:


```
{
  "meta":{
    "schema_version":"3.0",
    "api_version":"3.0",
    "message_type":"session_update",
    "stream_type":"session_all"
  },
  "content":{
    "group":{
      "first":[
        "Shop Maxymised"
```

```
        ],
        "last":[
          "Cart"
        ]
      }
    },
    "visitor":{
      "id":"1348447491483863571",
      "session":"1532364415416010076"
    }
}
```


Where Criteria - Optional

The WHERE criteria defines if/when a streams message is emitted. When an event is added to a session, it is compared against this criteria and if it evaluates to true, an update message is emitted.

If no WHERE clause is specified, every event that comes in to the session will result in an update message being emitted, as well as a closed message when the session is closed.

The ALL and the ANY operators are used if you want to evaluate all the events in the session, as opposed to just the current event being processed.

Note: Aliases and Select Critera are only used when sending messages, and cannot be used within the query in a WHERE (or HAVING) clause.

For example, this syntax is **NOT** supported:
Select data.wt.ti as 'A' where 'A" = 'B'
Select * where earliest(data.wt.ti) = 'B'
Select count(data.wt.ti) where count(data.wt.ti) >= 1
Select event_count() where event_count() >= 1

Type of Where criteria	Example	Behavior	Notes
Equals	data.wt.ti='title'	A message will be sent if the parameter exists, with the exact value specified.	Wild cards are not supported Case insensitive matching

Type of Where criteria	Example	Behavior	Notes
Exists	Exists(data.wt.ti)	A message will be sent if the parameter exists, regardless of value	Wild cards are not supported Case insensitive matching
Not Exists	NotExists(data.wt.ti) Not_Exists(data.wt.ti)	A message will be sent if the parameter does not exist	Wild cards are not supported Case insensitive matching
Not Equal	data.wt.ti!='title'	A message will be sent if the parameter doesn't exist, or doesn't have the exact value specified.	Wild cards are not supported Case insensitive matching
Like	data.wt.ti LIKE 't*'	A message will be sent if the parameter exists, and matches the value specified.	Case insensitive matching
Not Like	data.wt.ti NOT LIKE 't*' data.wt.ti NOT_LIKE 't*'	A message will be sent if the parameter doesn't exist, or it does not match the value specified.	Case insensitive matching
All	ALL(data.wt.cg_n = 'cg1')	A message will be sent if the parameter exists with the specified value on every event in the session	Case insensitive matching All events in session
Any	ANY(data.wt.cg_n = 'cg1')	A message will be sent if the parameter exists with the specified value on at least one event in the session	Case insensitive matching All events in session
Less Than	data.wt.tx_u < 3	A message will be sent if the parameter exists, and matches the value specified.	Numeric values only
Less Than or Equal To	data.wt.tx_u <= 3	A message will be sent if the parameter exists, and matches the value specified.	Numeric values only

Type of Where criteria	Example	Behavior	Notes
Greater Than	data.wt.tx_u > 3	A message will be sent if the parameter exists, and matches the value specified.	Numeric values only
Greater Than or Equal To	data.wt.tx_u >= 3	A message will be sent if the parameter exists, and matches the value specified.	Numeric values only
OR	data.wt.tx_u < 3 OR data.wt.ti = 'title'	A message will be sent if the parameter exists, and matches the value specified for either clause	
AND	data.wt.tx_u < 3 AND data.wt.ti = 'title'	A message will be sent if the parameter exists, and matches the value specified for all clauses	

Having Criteria - Optional

The HAVING clause defines which events in the session will be included, when querying session level data.

The HAVING clause is different from the WHERE clause, in that the HAVING clauses determines **WHICH** events should be included in the message, while the WHERE clause determines **IF** a message should be sent.

It is important to note that the HAVING clause is only applied to selections using the following list of session level Selectors:

Having Clause Selectors
CONCAT
CONCAT_DISTINCT
COUNT
COUNT_DISTINCT
EARLIEST
EARLIEST_INCLUDE_NULL
EVENT_COUNT()
LATEST
LATEST_INCLUDE_NULL

For example, given this select statement:

SELECT data.wt.ti AS 'current_page_title', CONCAT_DISTINCT(data.wt.ti, ',') AS 'all_phone_page_titles' HAVING data.wt.ti like '*phones*'

Since there is no WHERE clause, we will send out a message with every event.
The value of data.wt.ti will be sent as 'current_page_title' on every event, if present, regardless of whether or not it matches the HAVING clause. This is because a session-level selector is not being used here.
A session-level selector of CONCAT_DISTINCT is being used for 'all_phone_page_titles', so all distinct values from

every event that match the HAVING clause will also be included on every message.

For example:


```
{
  "meta":{
    "schema_version":"3.0",
    "api_version":"3.0",
    "message_type":"session_update",
    "stream_type":"session_all"
  },
  "all_phone_page_titles":"Google Phones,Samsung Phones",
  "current_page_title":"SLR Cameras"
}
```


If no HAVING clause is specified, every event in the session will be included when selecting session-level data.

The operators for HAVING are basically the same as WHERE, however you cannot use ALL or ANY in the HAVING clause. Also, just like the WHERE clause, Aliases and Select Criteria are not supported.

Type of Having criteria	Example	Behavior	Notes
Equals	data.wt.ti='title'	An event will be included in the selected data if the parameter exists with the exact value specified.	Wild cards are not supported Case insensitive matching
Exists	Exists(data.wt.ti)	An event will be included in the selected data if the parameter exists, regardless of value	Wild cards are not supported Case insensitive matching
Not Exists	NotExists(data.wt.ti) Not_Exists(data.wt.ti)	An event will be included in the selected data if the parameter does not exist	Wild cards are not supported Case insensitive matching

Type of Having criteria	Example	Behavior	Notes
Not Equal	data.wt.ti!='title'	An event will be included in the selected data if the parameter doesn't exist, or doesn't have the exact value specified.	Wild cards are not supported Case insensitive matching
Like	data.wt.ti LIKE 't*'	An event will be included in the selected data if the parameter exists, and matches the value specified.	Case insensitive matching
Not Like	data.wt.ti NOT LIKE 't*' data.wt.ti NOT_LIKE 't*'	An event will be included in the selected data if the parameter doesn't exist, or it does not match the value specified.	Case insensitive matching
Less Than	data.wt.tx_u < 3	An event will be included in the selected data if the parameter exists, and matches the value specified.	Numeric values only
Less Than or Equal To	data.wt.tx_u <= 3	An event will be included in the selected data if the parameter exists, and matches the value specified.	Numeric values only
Greater Than	data.wt.tx_u > 3	An event will be included in the selected data if the parameter exists, and matches the value specified.	Numeric values only
Greater Than or Equal To	data.wt.tx_u >= 3	An event will be included in the selected data if the parameter exists, and matches the value specified.	Numeric values only
OR	data.wt.tx_u >= 3 OR data.wt.ti = 'title'	An event will be included in the selected data if the parameter exists, and matches the value specified for either clause	
AND	data.wt.tx_u >= 3 AND data.wt.ti = 'title'	An event will be included in the selected data if the parameter exists, and matches the value specified for all clauses	