Last modified October 3, 2012 | **www.epsilon.com**

DREAM Real-Time Messaging Guide

Version 8.8.4

Copyrights

This documentation and related technology are governed by a user agreement and shall remain the sole and exclusive property of Epsilon. No part of this documentation or related technology may be used, reproduced, translated, displayed, distributed, disclosed, stored in a retrieval system or transmitted in any form or by any means without the written permission of Epsilon, unless otherwise stated in the user agreement. The information contained in this documentation is confidential and proprietary to Epsilon.

Disclaimer

Epsilon does not warrant, guarantee or make any representations or otherwise concerning the contents of this documentation or the applicability thereof. Epsilon reserves the right to change the contents of this document at any time without prior notification of such updates.

Trademarks

DREAM is a registered service mark of Epsilon.

CONTENT

CONTENT 1
Preface iv
Overview of Real Time Messaging 1
What is Real Time Messaging? 2
How Does RTM Work? 7
How Do I Implement RTM? 12
Creating and Updating Real-Time Messages 13
Enabling RTM in DREAM 14
Using DREAM to Create and Update RTM Messages 16
Sending Real-Time Messages 23
Using RTM Web Services Description Language 24
Available SOAP API Methods 34
Sending RTM Requests through SOAP 35
RTM API Authentication Strategy 36
Triggering an RTM Message 38
Creating RTMTriggerRequests 39
RTMTrigger Responses 47
Requesting an RTM Status Message 50
Creating RTMGetStatusRequests 51
RTMGetStatus Responses 53
Reporting on Real-Time Messages 57
RTM Reports in DREAM 58

PREFACE

About this guide

This guide is a part of the set of documents that describe the features of DREAM. This guide describes the Real Time Messaging (RTM) feature, DREAM's high-volume email solution that you can use to trigger highly personalized, transactional messages to your customers. These message include e-commerce transactions or responses to requests for information.

This guide explains how to use the RTM application programming interface (API) in conjunction with the DREAM application to create and send RTM messages.

Intended audience

This guide is intended for DREAM users who are responsible for designing and triggering DREAM messages in response to such user-initiated events. It is assumed that you have used DREAM and have a working knowledge of XML.

For more information about DREAM, refer to the online help, which can be accessed from the DREAM application.

How this guide is organized

This guide is divided into the following sections:

- Overview of Real Time Messaging explains what RTM is, how it works, and how you can create and send RTM messages.
- Creating and Updating Real-Time Messages describes how you can enable DREAM to send RTM messages and use the DREAM application to create and update RTM messages.
- Sending Real-Time Messages describes how you can use the XML-based RTM APIs to trigger real time messages and requests for the status of sent messages.
- Triggering an RTM Message includes a list of all the XML elements used to trigger RTM messages and an example of a request.
- Requesting an RTM Status Message includes a list of all the XML elements used to obtain the status of a sent RTM message and an examples of a status request.
- Reporting on Real-Time Messages explains how you can use DREAM's reporting function to monitor and analyze RTM messages.

Guide conventions

The following table lists the conventions used in this guide.

TABLE 1: Guide conventions

This...	Indicates...
Successive menu choices	Successive-menu choices are indicated with a greater-than sign (>) between the items that you select consecutively.
Bold text	This shows the names of menu items, dialog boxes, dialog box elements, and commands.
<variable name>	Variables that you must place in a text may appear between a greater-than and a lesser-than sign. When you type the command, replace this string with the relevant information. For example, for C:\Document and Settings\<your name>\Start Menu, John Smith might type C:\Document and Settings\JohnSmith\Start Menu.
`Courier text`	Code examples appear in courier text. It may represent text you type or data you read.
Note:	Notes contain additional useful information. Pay special attention to information highlighted this way.
Important:	Important text contains critical information.

Abbreviations

The following table lists the abbreviations used in this guide.

TABLE 2: Abbreviations

Abbreviation	Expansion
RTM	Real Time Messaging
WSDL	Web Services Description Language
SOAP	Simple Object Access Protocol

Overview of Real Time Messaging

This section describes Real Time Messaging (RTM) and how you can use this DREAM feature to send transactional and operational messages to your customers.

This section includes the following topics:

- What is Real Time Messaging?
- How Does RTM Work?
- How Do I Implement RTM?

What is Real Time Messaging?

RTM is a fault-tolerant, high-volume email solution that you can use to send highly personalized, HTML-rich, one-to-one transactional or operational emails to your customers in response to the transactions performed over the web, such as making an online purchase or requesting the status of an account.

This section includes the following topics:

- Types of RTM messages
- Overview of RTM
- Benefits of RTM
- RTM message content
- Fault tolerance
- Requirements
- Privacy
- Email address validation
- Delivery
- Reporting

Types of RTM messages

RTM can be used for various transactional and operational purposes, including:

- Order and service confirmations
- Reservation confirmations and electronic ticketing
- Billing and payment notices
- Notifications of updated profiles, password resets, and so on
- Welcome messages for program registration
- Responses to requests for information

Overview of RTM

The RTM process can be triggered when transactions are performed over the web, such as ecommerce transactions or automated notifications like stock or bank fraud alerts. These web transactions initiate a request to DREAM to deploy an RTM message containing the transaction-related information to your customer. The request to DREAM is generated using an XML request that you create to capture and pass information to DREAM for inclusion in the RTM message, as seen in Figure 1. For more information on this process, see How Does RTM Work?.

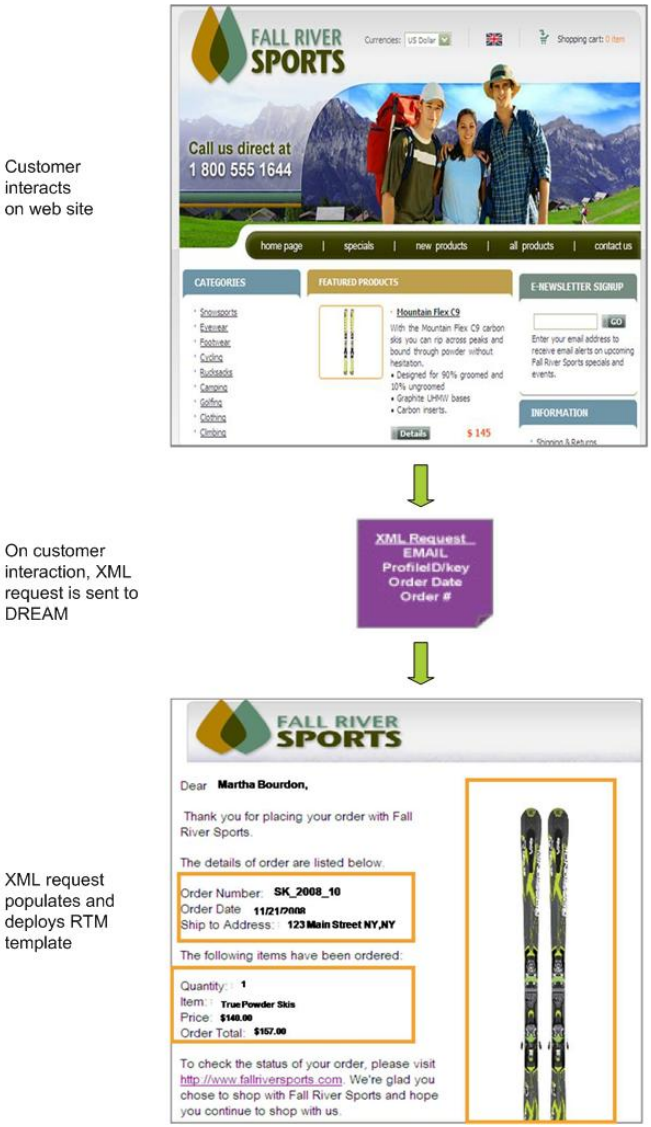

FIGURE 1: Overview of the RTM process

Benefits of RTM

Incorporating transactional and operational message deployment within your email marketing program offers distinct advantages over in-house or outsourced solutions. The use of a single tool to deploy all customer communications has the following benefits:

- Allows you to easily consolidate behavioral data
- Provides greater insight into revenue generation from all sources
- Takes advantage of Epsilon's delivery management tools and experience
- Allows inbox rates for RTM to mirror those of marketing communications

Because transactional and operational messages generally obtain the highest opens, RTM also allows you to fully engage a greater number of recipients through the use of rich HTML format content and consistent brand messaging.

RTM message content

RTM messages are highly customized, one-to-one communications with your customers. The information specific to the customer and the transaction on your web site, such as the client name, order number, payment amount received, or account status is passed to the RTM message using event variables. See Adding event variables for instructions on how to create event variables. Figure 2 is an illustration of an RTM message with the event variables highlighted.

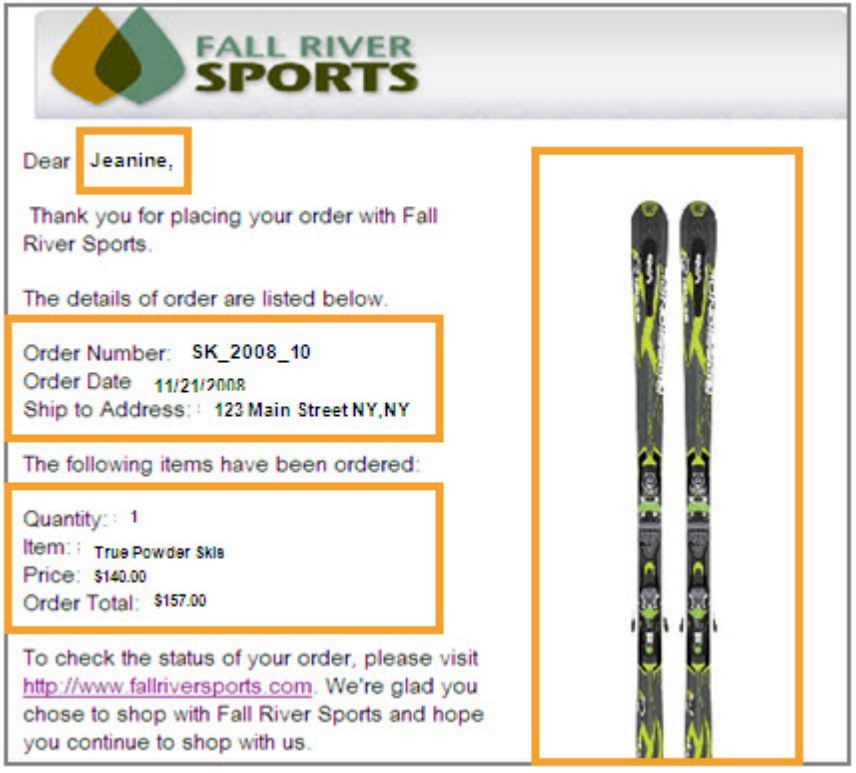

FIGURE 2: Sample RTM message

Fault tolerance

The DREAM RTM server does not require planned service outages. However, the use of dynamic personalization in the message and some message-tracking and reporting capabilities may be affected during DREAM maintenance. For more information, see When the DREAM database is unavailable.

Requirements

To support the high volume and rapid deployment of RTM messages, the following are required for RTM:

- Dedicated IP address
- Separate client in DREAM
- Unique profile list

Privacy

When RTM is used for transactional and operational purposes, the rules of CAN-SPAM and the usual measures taken to respect the privacy of subscribers are not always applicable. Use of a separate client and profile list help to ensure that you can deploy these types of messages to anyone interacting on your web site.

Note: For information on the application of the CAN-SPAM regulations to operational and transactional messages, we recommend reviewing the Federal Trade Commission's CAN-SPAM web page: http://www.ftc.gov/bcp/edu/pubs/business/ecommerce/bus61.shtm

Profile record optout flag

When the DREAM database is available, DREAM will verify the opt-out status of the intended recipient before the message is sent. If the optout flag in the profile record indicates the subscriber has opted out of your RTM profile list, the RTM message is suppressed.

Note: Using the available parameters, you can elect to bypass the opt-out setting and send the message, regardless of the recipient's opt-out status.

Email address validation

RTM validates the email addresses included in all RTM requests to ensure the addresses comply with the RFC 822 specification, a commonly-used standard defining the format for electronic messages. If the email address in the request is not RFC 822 compliant, the RTM message cannot be sent.

Note: DREAM RTM does not send emails to most role addresses. Role addresses are addresses used within an organization as internal email distribution lists, such as IT@, all@, or customerservice@.

Delivery

RTM messages follow the same delivery rules and logic as other message types deployed from DREAM. If an RTM message is sent to someone who is not on your

profile list, the recipient's email address and profile key are added to the profile list associated with the RTM template, however no additional information regarding the recipient is captured. If the email address is in a valid format but the mail is not being delivered, DREAM will continue to retry the message every 30 minutes for a total of 6 retires, at which time the email address is recorded as a soft bounce.

If the RTM message recipient is a member of a profile list, the message is delivered using the mailing preference (Text, HTML, Multipart) in the recipient's profile record. If the recipient is not in the profile list, a multipart message is deployed.

Reporting

You can use the reports in DREAM to monitor and analyze your RTM messages. Various reports in DREAM capture RTM mailing information, including:

- Summary Report
- Activity Download Report
- Executive Report
- Usage Summary Report
- Weekly Status Report

How Does RTM Work?

This section describes how RTM is designed and how RTM triggers messages in response to events and transactions on the web.

The topics in this section are as follows:

- How RTM processes trigger requests
- When the DREAM database is unavailable
- Components of the RTM architecture

How RTM processes trigger requests

RTM is designed to trigger messages in response to events and transactions that occur on your web site. When RTM receives an XML request through HTTP/HTTPS to trigger a message, it assigns a Transaction ID to the request and attempts to validate the request. If the request is validated, RTM processes the request's specifications, inserts the event variables into the RTM message, and sends the message to the customer. RTM then returns the Transaction ID to the requester for tracking purposes.

When the DREAM database is unavailable

As the information for the message is directly passed from your web site, RTM continues to populate event variables even when the DREAM database is unavailable; for example, during routine maintenance.

URL-tracking

URL-tracking in a message is accessible even when the DREAM database is unavailable. If the URLs have been tracked in previous messages (that is, the URLs have already been saved in DREAM), clicks on the URLs in the RTM message are tracked. However, if the URLs have not been tracked in other messages, they cannot be tracked when the DREAM database is unavailable.

DREAM reports

When the DREAM database becomes available again, DREAM reports on URL-tracking are updated in batch mode. Therefore, report data may not be updated as quickly as when DREAM functions normally.

Components of the RTM architecture

Figure 3 is a simplified view of the primary components of the RTM architecture.

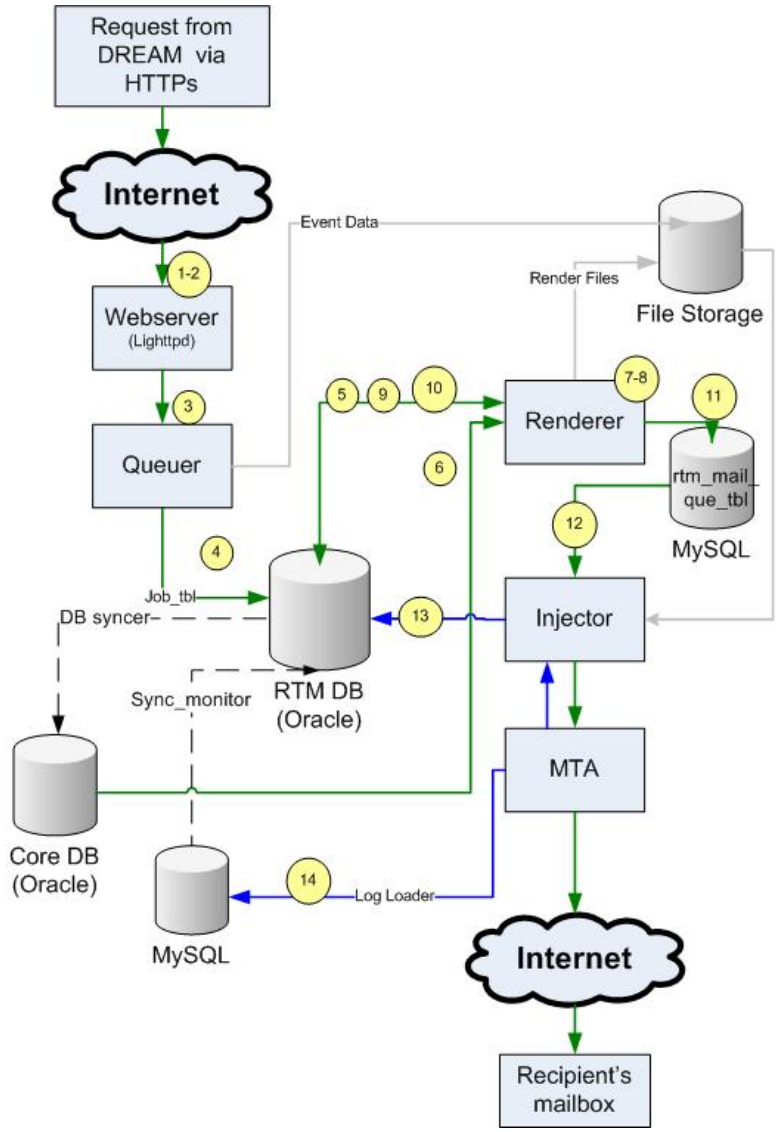

FIGURE 3: Primary components of the RTM architecture

TABLE 3: Primary RTM Components

Component	Description
Webserver	A computer program responsible for accepting HTTP requests and serving HTTP responses and optional data contents.
Queuer	A software component that accepts XML requests posted through HTTP/HTTPS to send RTM messages. The Queuer validates the requests, assigns a Transaction ID to each request, and returns that ID to the requester through HTTP/HTTPS.
Renderer	A software component that combines the requests from the Queuer into the template in the DREAM database and forwards the message to the Mailer.
Injector	A service that pushes multi-channel messages into the proper delivery service, for example SMTP for email.
MTA	The MTA (Mail Transfer Agent) takes the rendered content of the message and pushes it to the Internet for the recipient.

RTM Process Flow

HTTP Webserver

1 Receives the RTM request from the client

2 After successful authorization and validation, an RTM Trans_ID is assigned to the request

Queuer:

3 Receives the request parameters and queues the request for the renderer to process

4 Inserts the request parameters into the RTM Oracle database.

Renderer

5 Reads the request from the RTM database.

6 Retrieves data for the profile record of the recipient from the Oracle Client schema, if one exists

7 Compiles and personalizes the message based on profile and event data

8 Renders the mail with an RTM Tran_ID and profile_ID, if a profile record exists for the intended recipient

9 Inserts a status record into the Global Status Table of the RTM database indicting that the rendering is complete or has failed

10 If a profile record does not already exist, passes the client, profile list, and profile key information for the recipient to the RTM database. The database syncher then passes this information to the profile list associated with this client.

11 Generates the mail file to be mailed.

Injector

12 Retrieves the rendered content generated by the Renderer and pushes the content to the MTA

13 Inserts the final status of the send into the RTM database

14 Inserts the send activities (outbound SMTP) into mySQL database

MTA

14	Inserts the send activities (outbound SMTP) into mySQL database via the LogLoader
15	Sends the message to the intended recipient's mailbox

Note: Inbound SMTP, such as replies and bounces, are handled by the ReplyHander (not shown).

How Do I Implement RTM?

Before you can implement RTM in DREAM, you must first determine the kind of user-initiated actions that require RTM messages and define the content to be included in those messages. Once you have determined the actions and the content to be included in RTM, there are several DREAM components you can use:

- DREAM **application**: used to enable RTM and to create, update, and arm RTM messages. You can also use DREAM's reporting module to analyze the RTM data.
- **RTM API**: used to create XML requests that trigger RTM messages. In addition, the RTM API allows you to obtain the status of the RTM requests.

Implementation Process

Task	Description
Tasks performed outside of DREAM	
1	**Decide which transactions require RTM messages.** See What is Real Time Messaging?.
2	**Plan the content for the RTM messages.** Determine the event variables to be included in your RTM messages, along with the tracked URLs and message formats.
Tasks performed in DREAM	
1	Contact your Epsilon account representative to enable RTM.
2	Create RTM templates with the desired content.
3	Arm your RTM templates to be made available for RTM requests.
Tasks performed using the RTM API	
1	**Use the RTM API to create XML requests that trigger your RTM messages.** See Sending RTM Requests through SOAP.
2	**Create a program or web form that posts the XML requests.** See Sending RTM Requests through SOAP.
3	**Use the RTM API to check the status of the RTM requests.** You can check the status of an RTM message using either the Tracking ID or Transaction ID. Using the Tracking ID, DREAM limits the return to no more than 100 of the latest transactions. In some cases, even less than 100 are returned. Note: Tracking ID is not recommended for obtaining statuses; instead, use Transaction ID.

CREATING AND UPDATING REAL-TIME MESSAGES

To send RTM messages, you must first enable RTM in DREAM. Once enabled, you can create, update, and deploy your RTM messages.

This section discusses the following topics:

- Enabling RTM in DREAM
- Using DREAM to Create and Update RTM Messages

Enabling RTM in DREAM

Before you can create and trigger RTM messages, your new DREAM client must be enabled for RTM and users must be granted permission to create RTM templates. Contact your Epsilon account representative to request for enabling RTM.

Enabling the client for RTM

The RTM functionality is available only for RTM-enabled clients. Once RTM is enabled, RTM requests from the client are accepted and processed. If RTM is not enabled, any RTM request received is rejected.

Note: Only a user with administrative privileges can enable a client for RTM.

To enable a client to use RTM

Step	Action
1	Click **Admin** > **Client List**. The list of active clients is displayed.
2	Click the name of the client to enable RTM. The Edit Client Record screen is displayed.
3	Select the **Enable Real Time Messaging** check box.
4	Click **Submit**. The record is saved.

Assigning RTM privileges to users

The DREAM RTM functionality is available only to RTM-enabled clients. Access to the RTM functionality is granted at an individual user level within DREAM. There are two types of users who access the RTM functionality:

- **Client Users** – utilize the user interface; have permission to create, edit, enable or disable RTM messages for deployment; have the ability to review unique and aggregate reporting metrics for the emails that have deployed
- **Application Users** – profiles enabled to utilize the RTM AMI to trigger a RTM via XML HTTPS request; authorization to the user interface is denied for Application User profiles.

Note: User access can be given only to users for RTM-enabled clients.

To assign RTM privileges to a user

Step	Action
1	Click **Admin** > **User List**. The list of active users is displayed.
2	Select the client associated with the user from the **Client** list.
3	Click **Add** to create a new user. Alternatively, click **Edit** placed at the end of row for an existing user to access the user record. The Edit User Record screen is displayed.
4	Click the **User Type** list and select either of the following: **Client User**: if the user requires access to the DREAM user interface **Application User**: if the user requires access RTM only through the DREAM API
5	Provide the user information, including demographic information for the user, and so on.
6	Under the **Permissions** section, select the **API** check box. The RTM check box is automatically selected. Note: For the Application User type, only **API** option is displayed.
7	Repeat **Step 2** through **Step 4** for each user requiring RTM access.
8	Click **Submit**. The user record is saved.

Using DREAM to Create and Update RTM Messages

This section explains how to create and update RTM messages in DREAM. It includes the following topics:

- Creating an RTM message
- Adding event variables
- Personalizing URLs in RTM messages
- Arming RTM messages

Creating an RTM message

The process for creating an RTM message is similar to the process for creating a standard message in DREAM. RTM messages are created using an RTM template.

To create an RTM template

Step	Action
1	Click **Folders**. The Folder screen is displayed.
2	Double-click the name of the folder you want to open.. Any existing assets for the folder are displayed.
3	Click **New** on the toolbar and select **Template** from the list. The New Template screen is displayed.
4	Enter a **Name** for the template. **Name** is a mandatory field.
5	Enter a **Description** for the template.
6	Select a character set for this template from the **Charset** list.
7	Select a **Category** for the template from the list. **Category** is a mandatory field.
8	Select a **Profile List** for this template from the list.
9	Verify that the **From** email address for this template is correct. By default, the client **From** address is displayed. Any changes made to this address overrides the default.
10	Verify that the **Reply To** email address for this template is correct. By default, the client **Reply To** address is displayed. Any changes made to this address overrides the default.
11	Click **Real Time Template (RTM)** in **Type**.
12	Click **Create**. The Edit Template screen is displayed where you can specify the details for your RTM template.

Adding your RTM content

After you have created the RTM template, you are ready to add your RTM content using the Edit Template screen.

Step	Action
1	**Edit** the template.
2	Select **2: Content** in the **Step** drop-down box to edit your copy. The **Type** list is defaulted to ALL.
3	Select the **Type** from the drop down list: ALL, TEXT or HTML. This selection controls the display of the Text and HTML syntax boxes. Choosing TEXT or HTML will change the display to show only the selected format.
4	Type your text and HTML content, including the event variables, in the appropriate content fields. For additional information on event variables, see Adding event variables.As an alternative, you can also copy/paste the content from a file or upload the content file by selecting **Upload** from the **Action** list.
5	Click **Save and Check** to save your message content.
6	You can now arm your RTM template. For more information on arming the RTM template, see Arming RTM messages.

Adding event variables

The content of the RTM template must include the *event variables*. Event variables are the scripted content that acts as a placeholder for the transactional information passed from your web site through an XML request. For example, the customer's order information is passed and inserted into the template prior to deployment. The format of the event variables submitted within the XML post can be either plain text or can include fully functional HTML.

You must define the event variables that you want to pass in the RTM template. Event variables can be used to pass any subscriber data, including the customer's first name, address, or other information.

Event variables are defined using the following expression:

```
[+EVENT('VAR_NAME', 'Default value', 'Format')+]
```

TABLE 4: Elements of the Event Variable Expression

Element	Description
VAR_NAME	The name of the event variable to be displayed

Default value	The value to be displayed when the event variable value is unavailable
Format	The format for the variable when displayed in the template: • **InitCap**: Initial letter of the first word in the string is capitalized • **PropCap**: Initial letter in each word in the string is capitalized • **LCase**: All letters are lowercase • **UCase**: All letters are uppercase

Figure 4 displays the event variables in a sample order confirmation template and includes placeholders for the customer's first name, order number, order date, shipping address, an image of the product purchased, and so on.



FIGURE 4: DREAM template with event variables

If the data required for the RTM template cannot be made available using event variables but is stored in DREAM, you can use *reference lists* or *personalization* to

populate this information. However, the values stored in DREAM for personalization are not available for RTM during DREAM downtime, requiring the use of a default value. The information from the reference list is not affected by DREAM downtime however. In addition, because the individuals interacting with your web site may not be included in your profile list, this information may not be available for all RTM messages.

Note: Because individuals interacting with your web site may not be members of your profile list, the email address must be passed using an event variable.

When calling information from the reference list, use an event variable to link to the reference list value you want to include, such as:

```
[+REF_PROFILE('Ref_list1',EVENT('Product_ID'),'Prod_desc')+]
```

In the previous example, the product ID is passed to the RTM message. The reference list then provides the product description. For additional information on using reference lists and personalization, refer to the DREAM online help.

Important: Use personalization only when the information to be passed to the RTM message cannot be passed using event variables. When using personalization, always provide a default value to ensure that a value displays in your template during DREAM downtime.

Personalizing URLs in RTM messages

There are two ways to incorporate personalized URLs within your RTM message. You can either insert an event variable from the RTM request into a URL in the template or you can pass the personalized URL within the RTM request.

Inserting an event variable from the RTM request into a template URL:

Using PERL scripting, you can insert an event variable from the RTM request into a template URL. For example, the following link could be placed in the HTML format of the RTM template:

```
<a href="http://www.fallriversports.com/?id=[+EVENT('EV1',
'notfound:default',")+]">URL</a>
```

The RTM request would define a value for the event variable called "EV1":

```
<Variable>
    <Name>EV1</Name>
    <Value>28324</Value>
</Variable>
```

When received, the URL in the RTM would display the following link:

http://fallriversports.com/?id=28324

Note: Link tracking cannot be enabled using this personalization method.

Passing the personalized URL to the template as an event variable

You also have the option of passing the personalized URL to the template through the RTM request. Place the event variable for the personalized URL, such as `[+Event('EV1','default URL')+]`, within the RTM template where you would like the URL to display. When adding personalization to the HMTL content creative **do not** add double quotation marks at the beginning and end of the personalization brackets. An example of correct personalization within the html code is `<a href=[+EVENT('EV1','default URL',")+]>URL</a>.`

In the RTM request, define the personalized URL to be passed, for example:

```
<Variable>
    <Name>EV1</Name>
    <Value>http://www.fallriversports.com/?id=28324</Value>
</Variable>
```

Note: The usage of standard HTML code with event personalization to populate the entire URL will render an invalid URL. For example:

- If the personalization variable is `<a href="[+EVENT('EV1', 'default URL',")+]">`
- With a payload value of http://www.fallriversports.com for `EV1`.
- The **Expected Results for a RTM Message** would be a campaign deployed with the tracked link http://www.fallriversports.com.
- The **Actual Results for a RTM Message** is a campaign deployed with the **invalid** link "http://www.fallriversports.com".

Note: A default URL can be included in your request. A default URL displays when the personalized URL cannot be accessed.

This approach requires DREAM to trigger the RTM request to compile the personalized URL before generating the value for the event variable.

Enabling link tracking for URLs

If a URL is passed to the RTM template as an event variable (see above), link tracking can be enabled with the RTM request. To track your links, first define the event variable that will store the URL:

```
<Variable>
    <Name>EV1</Name>
    <Value>http://www.fallriversports.com/?id=28324</Value>
</Variable>
```

Next, define the URL that you want to track:

```
<TrackedUrls>
    <URL=”Home Page”>http://www.fallriversports.com/?id=28324</URL>
</TrackedUrls>
```

Important: The tracked link will not work if the RTM template contains PropCap, InitCap, UCase or LCase functions, for example <P>Event1=[+EVENT(EV1’,’not found:default’,’PropCap’)+]</P>

Arming RTM messages

An RTM template has a status of *armed* or *disarmed*. By default, a newly created RTM template is disarmed. You must arm the template to begin sending RTM messages. Once armed, the template cannot be edited until it is disarmed again. The status of the RTM template is displayed in the ARM/DISARM templates for Real Time Messaging section of the template, as seen in Figure 5.

Note: RTM templates must be armed to trigger messages to be deployed.

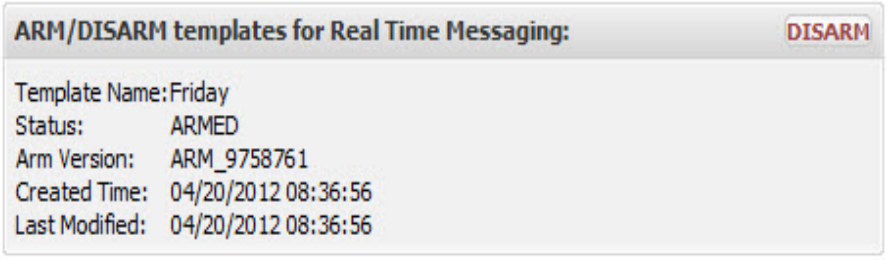

FIGURE 5: ARM/DISARM templates for Real Time Messaging section of the template

In addition to the template name and status (ARMED or DISARMED), this screen also displays the arm version number, created time and last modified time for the RTM template. The ArmVersion number is a unique identifier assigned by DREAM and changes depending on whether the template is armed, disarmed or re-armed.

To arm an RTM message

Step	Action
1	From the Edit Template screen, select **4: Real Time Messaging** from the **Step** drop-down box. The ARM/DISARM templates for Real Time Messaging section is displayed.
2	Click **Arm**. The message **Are you sure you want to arm the template?** is displayed. Note: By default, the template is in a DISARMED state.
3	Click **OK**. A message informing you that the template is being armed is displayed.
4	After the template has been armed, the status under the ARM/DISARM Templates for Real Time Messaging section is changed to **ARMED**. Note: You cannot edit the template unless it is **DISARMED**.

Note: You can preview the RTM template when it is armed, however you can only edit the template when it is in a disarmed status.

SENDING REAL-TIME MESSAGES

You must use the RTM Simple Object Access Protocol (SOAP) API to trigger an RTM message to an email address and to obtain the status information about your request. The RTM SOAP API is described using the Web Services Description Language (WSDL).

This section discusses the following topics regarding the RTM API:

- Using RTM Web Services Description Language
- Available SOAP API Methods
- Sending RTM Requests through SOAP
- RTM API Authentication Strategy

Using RTM Web Services Description Language

WSDL location

The WSDL document can be accessed from the following locations:

Dallas server: https://rtm.dls.dream.epsilon.com/rtmsd?wsdl

New Jersey server: https://rtm.nj.dream.epsilon.com/rtmsd?wsdl

WSDL code for RTM

Following is the WSDL code for RTM:

```
<?xml version="1.0"?>
<wsdl:definitions
    xmlns:tns="DREAMRTMReq.wsdl"
    xmlns:rtmreq="DREAMRTMWebletReq.xsd"
    xmlns:rtmresp="DREAMRTMResp.xsd"
    xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    targetNamespace="DREAMRTMReq.wsdl">
    <wsdl:types>
        <xsd:schema targetNamespace="DREAMRTMWebletReq.xsd"
        xmlns:xsd="http://www.w3.org/2001/XMLSchema">
            <xsd:annotation>
                <xsd:documentation>
                    RTM Request Schema
                </xsd:documentation>
            </xsd:annotation>
            <!-- Define the type TVariable -->
            <xsd:complexType name="TVariable">
                <xsd:sequence>
                    <xsd:element name="Name" type="xsd:string"
                    minOccurs="1" maxOccurs="1"/>
                    <xsd:element name="Value" type="xsd:string"
                    minOccurs="1" maxOccurs="1"/>
                </xsd:sequence>
            </xsd:complexType>
            <!-- Define the type TEventVariables for DREAM -->
            <xsd:complexType name="TEventVariables">
                <xsd:sequence>
                  <xsd:element name="Variable" type="rtmreq:TVariable"
                  minOccurs="1" maxOccurs="unbounded"/>
                </xsd:sequence>
            </xsd:complexType>
            <!-- Define the type TAttribute for DREAM -->
            <xsd:complexType name="TAttribute">
                <xsd:sequence>
                    <xsd:element name="Key" type="xsd:string"
```

```
            minOccurs="1" maxOccurs="1"/>
            <xsd:element name="Value" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <!-- Define the type TProfileAttributes for DREAM -->
    <xsd:complexType name="TProfileAttributes">
        <xsd:sequence>
            <xsd:element name="Attribute"
            type="rtmreq:TAttribute" minOccurs="1"
            maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TURL">
        <xsd:simpleContent>
            <xsd:extension base="xsd:string">
                <xsd:attribute name="label" type="xsd:string"
                use="optional"/>
            </xsd:extension>
        </xsd:simpleContent>
    </xsd:complexType>
    <!-- Define the type TDREAMTrackedURLs for DREAM -->
    <xsd:complexType name="TDREAMTrackedURLs">
        <xsd:sequence>
            <xsd:element name="URL" type="rtmreq:TURL"
            minOccurs="1" maxOccurs="unbounded"/>
        </xsd:sequence>
    </xsd:complexType>
    <xsd:complexType name="TEmailProfile">
        <xsd:sequence>
            <xsd:element name="EmailAddress" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
            <xsd:element name="ProfileKey" type="xsd:string"
            minOccurs="1" maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <!-- Define the type TTriggerEventEmailAddress for DREAM -->
    <xsd:complexType name="TTriggerEventEmailAddress">
        <xsd:sequence>
            <xsd:choice>
                <xsd:element name="EmailAddress"
                type="xsd:string"/>
                <xsd:element name="ProfileKey" type="xsd:string"/
                >
                <!--
                  <EmailProfile> element allows to specify both
                    EmailAddress and ProfileKey
                -->
```

```
                <xsd:element name="EmailProfile"
                type="rtmreq:TEmailProfile"/>
            </xsd:choice>
            <xsd:element name="FromEmailAddress"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
          <xsd:element name="FromDisplayName" type="xsd:string"
          minOccurs="0" maxOccurs="1"/>
            <xsd:element name="ReplyToEmailAddress"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="ReplyToDisplayName"
            type="xsd:string" minOccurs="0" maxOccurs="1"/>
            <xsd:element name="EventVariables"
            type="rtmreq:TEventVariables" minOccurs="0"
            maxOccurs="1"/>
            <xsd:element name="ProfileAttributes"
            type="rtmreq:TProfileAttributes" minOccurs="0"
            maxOccurs="1"/>
            <xsd:element name="TrackedUrls"
            type="rtmreq:TDREAMTrackedURLs" minOccurs="0"
            maxOccurs="1"/>
        </xsd:sequence>
    </xsd:complexType>
    <!--
       Element <Defer> encapsulates a choice between the number
       minutes or date and time when the request will expire
    -->
    <xsd:complexType name="TDefer">
        <xsd:choice>
            <xsd:element name="MaximumDeferrableMinutes"
            type="xsd:unsignedInt"/>
            <xsd:element name="MaximumDeferrableDateTime"
            type="xsd:dateTime"/>
        </xsd:choice>
    </xsd:complexType>
    <xsd:complexType name="TToEmailAddress">
        <xsd:sequence>
            <xsd:element name="EventEmailAddress"
            type="rtmreq:TTriggerEventEmailAddress"/>
          <!-- client-provided tracking id (may be not unique),
          which can be used later on to request the status
          through RTM status request -->
            <xsd:element name="TrackingID" type="xsd:string"
            minOccurs="0" maxOccurs="1"/>
            <!--ByPassOptOut is 0 by default, which means "not
            bypassing opt-out subscriber" -->
           <xsd:element name="ByPassOptOut" type="xsd:boolean"
            minOccurs="0" maxOccurs="1" default="0"/>
        </xsd:sequence>
    </xsd:complexType>
```

```
<!--
            Declare DREAM RTM Trigger request
           TriggerEmailTarget
-->
<xsd:complexType name="TTriggerEmailTarget">
    <xsd:sequence>
        <xsd:element name="Folder" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
        <xsd:element name="Template" type="xsd:string"
        minOccurs="1" maxOccurs="1"/>
        <xsd:element name="Defer" type="rtmreq:TDefer"
        minOccurs="0" maxOccurs="1"/>
        <!--ActionAfterDeferredTime is 0 by default, which
        means "send a request anyway" -->
        <xsd:element name="ActionAfterDeferredTime"
        type="xsd:boolean" minOccurs="0" maxOccurs="1"
        default="0"/>
        <!--
            <ToEmailAddress> elements can be included more
             than once to specify multiple destinations for
             the same template
        -->
        <xsd:element name="ToEmailAddress"
        type="rtmreq:TToEmailAddress" minOccurs="1"
        maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<!--
          Declare type of the main element of DREAM trigger
          request
-->
<xsd:complexType name="TRTMTriggerTemplate">
    <xsd:sequence>
        <xsd:element name="TriggerEmailTarget"
        type="rtmreq:TTriggerEmailTarget" minOccurs="1"
        maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TIDs">
    <xsd:sequence>
        <xsd:element name="ID" type="xsd:string"
         minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<!-- enumeration of ID types -->
<xsd:simpleType name="TType">
    <xsd:restriction base="xsd:string">
        <xsd:enumeration value="TransactionID"/>
        <xsd:enumeration value="TrackingID"/>
```

```
    </xsd:restriction>
 </xsd:simpleType>
 <xsd:complexType name="TTransaction">
     <xsd:sequence>
        <!-- <IDs> element includes an array of IDs to query
         -->
         <xsd:element name="IDs" type="rtmreq:TIDs"
         minOccurs="1" maxOccurs="1"/>
         <!-- type of IDs to query (TransactionID or
         TrackingID) -->
         <xsd:element name="Type" type="rtmreq:TType"
         minOccurs="0" maxOccurs="1"
         default="TransactionID"/>
     </xsd:sequence>
 </xsd:complexType>
 <!--
            Declare the type of the main element of RTM Get
            Status request for DREAM
 -->
 <xsd:complexType name="TRTMGetStatus">
     <xsd:sequence>
         <xsd:element name="Transactions"
         type="rtmreq:TTransaction" minOccurs="1"
         maxOccurs="1"/>
     </xsd:sequence>
 </xsd:complexType>
 <!-- enumeration of Hash types -->
 <xsd:simpleType name="THash">
     <xsd:restriction base="xsd:string">
         <xsd:enumeration value="MD5"/>
         <xsd:enumeration value="SHA1"/>
         <xsd:enumeration value="SHA256"/>
     </xsd:restriction>
 </xsd:simpleType>
<!-- this type encapsulates authentication parameters of all
message types -->
 <xsd:complexType name="Headers">
     <xsd:sequence>
         <xsd:element name="Partner" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Client" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Timestamp" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Username" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Password" type="xsd:string"
          minOccurs="1" maxOccurs="1"/>
         <xsd:element name="Hash" type="rtmreq:THash"
```

```
                minOccurs="0" maxOccurs="1" default="SHA256"/>
            </xsd:sequence>
        </xsd:complexType>
        <xsd:complexType name="TRTMTriggerRequest">
            <xsd:sequence>
                <xsd:element name="Header" type="rtmreq:Headers"
                minOccurs="1" maxOccurs="1"/>
                <xsd:element name="RTMTriggerTemplate"
                type="rtmreq:TRTMTriggerTemplate" minOccurs="1"
                maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
        <!-- main element of RTMTriggerRequest message -->
        <xsd:element name="RTMTriggerRequest"
        type="rtmreq:TRTMTriggerRequest"/>
        <!-- declare type of the main Get Status Request element -->
        <xsd:complexType name="TRTMGetStatusRequest">
            <xsd:sequence>
                <xsd:element name="Header" type="rtmreq:Headers"
                minOccurs="1" maxOccurs="1"/>
                <xsd:element name="RTMGetStatus"
                type="rtmreq:TRTMGetStatus" minOccurs="1"
                maxOccurs="1"/>
            </xsd:sequence>
        </xsd:complexType>
        <!-- -->
      <!-- Declare the main element of RTMGetStatusRequest  message
      -->
        <!-- -->
        <xsd:element name="RTMGetStatusRequest"
         type="rtmreq:TRTMGetStatusRequest"/>
    </xsd:schema>
    <!-- namespace describing response elements -->
    <xsd:schema targetNamespace="DREAMRTMResp.xsd" xmlns:xsd="http:/
     /www.w3.org/2001/XMLSchema">
        <!-- Define the type TRecipientDetail -->
        <xsd:complexType name="TRecipientDetail">
            <xsd:sequence>
               <xsd:element name="EmailAddress" type="xsd:string"/>
                <xsd:element name="Status" type="xsd:integer"/>
                <!-- human readable description of the Status -->
                <xsd:element name="Description" type="xsd:string"/>
                <!--
                    Last time the job had some activity. If the job
                    was just received and no other activity happened
                    yet, then it will not be included
                -->
               <xsd:element name="ProcessedTime" type="xsd:string"
```

```
        minOccurs="0" maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="TForRTMEmailToEmailAddresses">
    <xsd:sequence>
       <xsd:element name="ReceivedTime" type="xsd:string"/>
        <xsd:element name="RecipientProcessingDetail"
        type="rtmresp:TRecipientDetail"/>
    </xsd:sequence>
</xsd:complexType>
<!-- Define the type TTransactionStatus -->
<xsd:complexType name="TTransactionStatus">
    <xsd:sequence>
        <xsd:element name="TransactionID"
        type="xsd:positiveInteger"/>
        <!-- only included if TrackingID exists for this
        TransactionID -->
        <xsd:element name="TrackingID" type="xsd:string"
        minOccurs="0" maxOccurs="1"/>
        <xsd:element name="ForRTMEmailToEmailAddresses"
        type="rtmresp:TForRTMEmailToEmailAddresses"/>
    </xsd:sequence>
</xsd:complexType>
<!--
            Declare TTransactionResult type
-->
<xsd:complexType name="TTransactionResult">
    <xsd:sequence>
        <!-- result code for this transaction ID -->
        <xsd:element name="Result" type="xsd:integer"/>
      <!-- human readable representation of the Result code
      -->
        <xsd:element name="Description" type="xsd:string"/>
        <!-- if error occurred then TransactionID is not
        returned -->
        <xsd:element name="TransactionID"
        type="xsd:positiveInteger" minOccurs="0"
        maxOccurs="1"/>
       <!-- TrackingID is returned only if it was provided
        in the original trigger mail request -->
        <xsd:element name="TrackingID" type="xsd:string"
        minOccurs="0" maxOccurs="1"/>
        <xsd:element name="ProfileKey" type="xsd:string"/>
    </xsd:sequence>
</xsd:complexType>
<!-- global result code for this response -->
<xsd:element name="Code" type="xsd:integer"/>
<!-- human readable representation of the Code -->
```

```
<xsd:element name="Description" type="xsd:string"/>

<xsd:complexType name="TTriggerResultData">
    <xsd:sequence>
        <xsd:element name="TransactionResult"
        type="rtmresp:TTransactionResult" minOccurs="1"
        maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>

<xsd:complexType name="TGetStatusResultData">
    <xsd:sequence>
        <xsd:element name="TransactionStatus"
        type="rtmresp:TTransactionStatus" minOccurs="1"
        maxOccurs="unbounded"/>
    </xsd:sequence>
</xsd:complexType>
<!-- declare type of main element of Trigger Response -->
<xsd:complexType name="TTriggerResponse">
    <xsd:sequence>
        <!-- global result code for this response -->
        <xsd:element ref="rtmresp:Code"/>
        <!-- human readable representation of the Code -->
       <xsd:element ref="rtmresp:Description" minOccurs="0"
       maxOccurs="1"/>
        <xsd:element name="ResultData"
        type="rtmresp:TTriggerResultData" minOccurs="0"
        maxOccurs="1"/>
    </xsd:sequence>
</xsd:complexType>
<!--                                                          -->
<!-- Declare and define the main element of TriggerResponse
-->
<!--                                                          -->
 <xsd:element name="TriggerResponse"
 type="rtmresp:TTriggerResponse"/>
<!-- declare type of main element of Get Status Response -->
 <xsd:complexType name="TGetStatusResponse">
    <xsd:sequence>
        <!-- global result code for this response -->
        <xsd:element ref="rtmresp:Code"/>
        <!-- human readable representation of the Code -->
       <xsd:element ref="rtmresp:Description" minOccurs="0"
       maxOccurs="1"/>
        <xsd:element name="ResultData"
        type="rtmresp:TGetStatusResultData" minOccurs="0"
        maxOccurs="1"/>
    </xsd:sequence>
```

```
            </xsd:complexType>
            <!-- -->
          <!-- Declare and define the main element of GetStatusResponse
          -->
            <!-- -->
            <xsd:element name="GetStatusResponse"
            type="rtmresp:TGetStatusResponse"/>
        </xsd:schema>
    </wsdl:types>
    <wsdl:message name="RTMTriggerRequest">
        <wsdl:part name="body" element="rtmreq:RTMTriggerRequest"/>
    </wsdl:message>
    <wsdl:message name="RTMGetStatusRequest">
        <wsdl:part name="body" element="rtmreq:RTMGetStatusRequest"/>
    </wsdl:message>
    <wsdl:message name="RTMTriggerResponse">
        <wsdl:part name="body" element="rtmresp:TriggerResponse"/>
    </wsdl:message>
    <wsdl:message name="RTMGetStatusResponse">
        <wsdl:part name="body" element="rtmresp:GetStatusResponse"/>
    </wsdl:message>
    <wsdl:portType name="RTMRequestPortType">
        <wsdl:operation name="MakeTriggerRequest">
            <wsdl:input message="tns:RTMTriggerRequest"/>
            <wsdl:output message="tns:RTMTriggerResponse"/>
        </wsdl:operation>
        <wsdl:operation name="MakeGetStatusRequest">
            <wsdl:input message="tns:RTMGetStatusRequest"/>
            <wsdl:output message="tns:RTMGetStatusResponse"/>
        </wsdl:operation>
    </wsdl:portType>
    <wsdl:binding name="RTMBinding" type="tns:RTMRequestPortType">
        <soap:binding style="document" transport="http://
        schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="MakeTriggerRequest">
            <soap:operation soapAction="urn:#MakeTriggerRequest"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="MakeGetStatusRequest">
            <soap:operation soapAction="urn:#MakeGetStatusRequest"/>
            <wsdl:input>
                <soap:body use="literal"/>
            </wsdl:input>
```

```
            <wsdl:output>
                <soap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="RTMService">
        <wsdl:documentation>RTM service</wsdl:documentation>
        <wsdl:port name="RTMRequestPort" binding="tns:RTMBinding">
            <!--
                Below is the URL of the Dallas server RTM Web Services
                interface. Use “https://rtm.nj.epsilon.com/rtmsd/” for
                the NJ RTM Web Services interace.
            -->
            <soap:address location="https://rtm.dls.dream.epsilon.com/
            rtmsd"/>
        </wsdl:port>
    </wsdl:service>
</wsdl:definitions>
```

Available SOAP API Methods

The RTM SOAP API described in the RTM WSDL contains two SOAP API methods:

- MakeTrigger Requests
- MakeGetStatus Requests

MakeTrigger Requests

The MakeTriggerRequest method is used to trigger an RTM message to the recipient and a response to the sender indicating the message was successfully sent or failed to be delivered. The MakeTriggerRequest consists of two SOAP messages as listed in the following table:

TABLE 5: SOAP messages of the MakeTriggerRequest method

Method	Description
RTMTriggerRequest	Request sent to trigger the RTM message
RTMTriggerResponse	Received as a response to the RTMTriggerRequest message

Information on using the MakeTriggerRequest to create an RTM message is provided in Creating RTMTriggerRequests section of this guide.

MakeGetStatus Requests

The MakeGetStatusRequest method can be used to trigger a request to DREAM to return the status of a specific RTM message based on either the message Transaction ID or Tracking ID. The MakeGetStatusRequest consists of two SOAP messages as listed in the following table:

TABLE 6: SOAP messages of the MakeGetStatusRequest method

Method	Description
RTMGetStatusRequest	Request sent to obtain the status of a specific RTM message
RTMGetStatusResponse	Received as a response to the RTMGetStatusRequest message

Information on using the MakeGetStatusRequest to obtain the status of an RTM message is provided in Creating RTMGetStatusRequests section of this guide.

Sending RTM Requests through SOAP

After you create a trigger or status request using one of the RTM APIs, you must post the request through SOAP to send the RTM message or verify the status of a sent message.

You can use any programming language that supports SOAP to create the request. Create request/response objects generated by language-specific compilers from the WSDL description of the RTM interface in your code. For additional information on the WSDL, including its location, see the Using RTM Web Services Description Language section of this guide.

Note: RTM requests fail when the request contains wide UTF-8 encoded data. UTF-8 encoded data is considered wide if the data is greater than one (1) byte.

Complete the mandatory fields of the request object, along with any optional fields needed. Once the code is created, call one of the two available RTM interface methods: MakeTriggerRequest or MakeGetStatusRequest. A response object contains the result of the call to RTM.

Your language-specific SOAP library should handle all low-level details such as making TCP connection to the RTM system, encapsulating a SOAP XML message inside HTTP request, and receiving HTTP response and populating the response object with values from the response XML message.

SOAP-specific errors are reported as appropriate for the chosen language (exceptions or error return codes). RTM-specific errors are reported inside the response object by means of its Code and Description elements.

Note: For working samples in Perl, C#, Java, and Python, log on to the Epsilon Learning Center (http://learn.epsiloninteractive.com) and download the SOAP sample files. Click the **Browse Resources** link in the left-navigation pane, and then click the **DREAM 8** tab at the top of the page. Contact your account representative if you need the registration key for the web site.

RTM Web Services URL

Following are the URLs for the Dallas and New Jersey server with DREAM RTM Web Services:

Dallas server: https://rtm.dls.dream.epsilon.com/rtmsd/

New Jersey server: https://rtm.nj.dream.epsilon.com/rtmsd/

RTM API Authentication Strategy

Real Time Messages can be initiated once a user has been authenticated by the Epsilon HTTPS web server. Epsilon's web server authentication strategy is a two-step authorization process.

Step 1: Validation of SOAP request header

- Partner/Client
  - Validation of DREAM partner and client fields within the request. The partner and client values will correspond to your client instance within the User Interface.
- User Name
  - Validate the user name provided exists as an application user within the partner and client previously validated.
- User Password Hex Conversion

  Each application user profile contains a case-sensitive password. Every transaction requires the submission of the password containing a prepending timestamp submitted in HEX hashed format creating a unique password for each transaction.
  - Standard password requirements include:
  - The system shall enforce a minimum password length of eight (8) characters.
  - Passwords shall enforce rules to ensure that there are no more than three repeating characters.
  - Passwords shall contain at least one alpha and one numeric character.
  - There shall be a mechanism to ensure that user-ids and passwords do not contain inappropriate words such as webmaster, administrator, etc.
  - The system shall use password history to prevent users from reusing previous passwords for a minimum of 6 occurrences.
  - The system shall ensure that there is a mechanism in place to assure that passwords generated on behalf of an individual user conform to password management standards.
  - The system shall ensure the password does not contain the login name spelled forward or backward.

Step 2: Validation of SOAP request body

- Profile list name
  - Validate the profile list name exists within the partner and client previously validated.
- Folder name
  - Validate the folder name exists within the partner and client previously validated.
- Template name
  - Validate the template name exists within the partner, client, and folder name previously validated. Also validate the template is enabled for deployment.

Term	Description
Partner	A collection of different business lines for an organization that share common properties in DREAM. This is the highest level in the DREAM hierarchy.
Client	Delineates a line of business, product or service, or the like under the partner. Clients can represent any type of business, such as a website, a publication, or a brick and mortar business.
Profile List	A collection of subscriber profiles maintained in DREAM to which email messages are mailed.
Folder	A location in DREAM that houses all of the assets related to a specific campaign.
Template	An email marketing message.

API Encryption/ SSL token

Communications between the client application and the Real Time Message API can be secured by using the Secure Sockets Layer (SSL) protocol for HTTP requests and responses. The SSL certificate consists of a public key and a private key. The public key is used to encrypt information and the private key is used to decipher it. When a web browser points to a secured domain, a level of encryption is established based on the type of SSL certificate as well as the client web browser, operating system and host server's capabilities.

This recommended approach for securing Internet-based communications was designed to prevent eavesdropping and message tampering.

TRIGGERING AN RTM MESSAGE

The MakeTriggerRequest API is used to trigger a real time message from DREAM to the intended recipient. For each RTM message requested, a response is also returned to the sender indicating the status of the RTM message, for example whether it was successfully delivered or failed to deliver and why.

This section discusses the following topics regarding the MakeTriggerRequest API:

- Creating RTMTriggerRequests
- RTMTrigger Responses

Creating RTMTriggerRequests

The RTMTriggerRequest call to DREAM is used to trigger an RTM message. Each RTMTriggerRequest sent to DREAM generates two messages, as seen in the table below.

TABLE 7: Messages triggered by an RTMTriggerRequest

Message	Description
RTM message	The message deployed to the recipient when an RTMTriggerRequest call is received by DREAM
RTMTriggerResponse	The response returned to the sender regarding the status of the RTM message, for example whether the transaction was successfully completed or failed to be delivered

This section of guide describes both the RTMTriggerRequest and the response to the request, including:

- RTMTrigger Request elements
- RTMTrigger Request sample code

RTMTrigger Request elements

The following table lists the elements you can use in the SOAP message to trigger an RTM message.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
`<RTMTriggerRequest>`	Root element. Contains all the information associated with triggering RTM messages.	No nesting required; this is the root element.
`<Header>`	Specifies validation and authentication parameters of the message.	Nested within the <RTMTriggerRequest> element. Mandatory.
`<Partner>`	DREAM Partner name Format = String Max. character length = 60	Nested within the <Header> element. Mandatory.
`<Client>`	DREAM Client name Format = String Max. character length = 60	Nested within the <Header> element. Mandatory.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
<Timestamp>	Current time in YYYY-MM-DDTHH:MM:SS.SSS format with millisecond precision (in UTC). Format = Date Max. character length = 23	Nested within the <Header> element. Mandatory.
<Username>	Username provided by Epsilon for RTM HTTP requests. Username = String Max. character length = 20	Nested within the <Header> element. Mandatory.
<Password>	Password, which is in HEX format and calculated by prepending the timestamp to the actual password and calculating the hash of the resulting string.	Nested within the <Header> element. Mandatory.
<Hash>	Hashing algorithm. Accepted values: MD5, SHA1, SHA256. If not specified, SHA256 is used.	Nested within the <Header> element. Optional.
<RTMTriggerTemplate >	Specifies an RTM message to trigger.	Nested within the RTMTriggerRequest element. Mandatory.
<TriggerEmailTarget>	Specifies RTM message target and other information.	Nested within the RTMTriggerTemplate element. Mandatory.
<Folder>	Folder name where the message was created. Format = String Max. character length = 60	Nested within the TriggerEmailTarget element. Mandatory.
<Template>	Name of the template. Format = String Max. character length = 60	Nested within the TriggerEmailTarget element. Mandatory.
<Defer>	Specifies parameters used to set the maximum time before which the template must be sent. If you do not include this element, the message can be sent at any time.	Nested within the TriggerEmailTarget element. Optional.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
`<MaximumDeferrable Minutes>`	Sets the maximum time (in minutes) for the RTM message to be sent. If the message is not sent in the specified time, the action specified in ActionAfterDeferredTime is executed. Format = String Max. character length = 1 Note: You cannot use MaximumDeferrable Minutes and MaximumDeferrableDate Time in the same request.	Nested within the Defer element. Optional.
`<MaximumDeferrable DateTime>`	Sets the expiration date and time before which the message must be sent: • `YYYY-MM-DDTHH24:MM:SSZ` is interpreted as UTC, and not local time. For example, 2004-07-01T14:40:10Z is 2.40 P.M. GMT. • `YYYY-MM-DDTHH24:MM:SS+/-hh:mm` is interpreted as local time, where the +/- hh:mm is the time difference from the local to the UTC time. For example, +05:00 for EST. The MaximumDeferrable DateTime must be greater than the current date and time. If you do not include this element, the message can be sent at any time. If the message does not get sent before the specified time, the action specified in ActionAfterDeferredTime is executed. Note: You cannot use MaximumDeferrableDate Time and MaximumDeferrable Minutes in the same request.	Nested within the Defer element. Optional.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
<ActionAfterDeferredTime>	Specifies the action taken if the message is not sent before the expiration time. 0: Send the RTM message anyway. (This is the default action.) 1: Do not send the message. Format = String Max. character length = 1	Nested within the TriggerEmailTarget element. Optional.
<ToEmailAddress>	Contains the elements you can use to specify the email address to send the RTM message to, create event variables, and specify Profile Attributes and the URLs to track. Also, you can specify an optional Tracking ID identifier.	Nested within the TriggerEmailTarget element. Mandatory, one element per destination email address.
<ByPassOptOut>	Specify whether the message should bypass the opt-out status of the intended recipient. 0 - Do not bypass the opt-out status (default) 1 - Bypass the opt-out status	Nested within ToEmailAddress element. Optional.
<EventEmailAddress>	Contains the elements you can use to specify the email address to which the RTM message is sent.	Nested within the ToEmailAddress element. Mandatory.
<TrackingID>	Contains an arbitrary label for the specific email that can be used later to get transaction statuses.	Nested within the ToEmailAddress element. Optional.
<EmailAddress>	Email address to which the RTM message should be sent. Format = String Max. character length = 100	Nested within the EventEmailAddress element. Mandatory.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
<ProfileKey>	Profile key of the recipient to whom the RTM message should be sent.	The EmailProfile element is nested within the EventEmailAddress element. Sample code is: <EventEmailAddress> <EmailProfile> <EmailAddress>xzhu@epsilon.com</EmailAddress> <ProfileKey>99</ProfileKey> </EmailProfile> </EventEmailAddress>
<EmailProfile>	Contains email address and the profile key of the recipient of the RTM message. Note: A maximum of one element of <EmailAddress>, <ProfileKey>, or <EmailProfile> can be used to specify the destination of the RTM message. If <EmailProfile> is specified, it must contain both email address and profile key inside <EmailAddress> and <ProfileKey> child elements.	
<FromEmailAddress>	Overrides the default From Address of the message. Format = String Max. character length = 100	Nested within the EventEmailAddress element. Optional.
<FromDisplayName>	Overrides the default From Display Name of the message. Format = String Max. character length = 100	Nested within the EventEmailAddress element. Optional.

TABLE 8: RTMTriggerRequest elements

Element Name	Description	Nesting
<ReplyToEmailAddress>	Overrides the default Reply-To Address of the message. Format = String Max. character length = 100	Nested within the EventEmailAddress element. Optional.
<ReplyToDisplayName>	Overrides the default Reply-To Display Name of the message. Format = String Max. character length = 100	Nested within the EventEmailAddress element. Optional.
<EventVariables>	Contains the elements you use to include event variables referenced in the RTM message.	Nested within the EventEmailAddress element. Optional.
<Variable>	Contains the elements that specify the event variable.	Nested within the EventVariables element. Must be specified for each event variable.
<Name>	Event variable name.	Nested within the Variable element. Mandatory.
<Value>	The content of the event variable.	Nested within the Variable element. Mandatory.
<ProfileAttributes>	Contains elements you can use to specify override values of profile attributes in the RTM message.	Nested in the EventEmailAddress element. Optional.
<Attribute>	Contains the elements that specify name and value of a profile attribute. Must be specified once per profile attribute you want to override.	Nested in the ProfileAttributes element. Mandatory.
<Key>	Profile attribute name that you want to override.	Nested in the Attribute element. Mandatory.
<Value>	Profile attribute value you want to override with.	Nested in the Attribute element. Mandatory.
<TrackedUrls>	Contains URLs to be tracked.	Nested within the EventEmailAddress element. Optional.
<URL>	The URL to be tracked inside event variables.	Nested within the TrackedUrls element. Mandatory.

RTMTrigger Request sample code

The following is an example of an RTMTriggerRequest code (not including the SOAP envelope). The current size limit for an RTMTrigger file is 1MB.

```
<rtmreq:RTMTriggerRequest xmlns:rtmreq="DREAMRTMWebletReq.xsd">
   <Header>
    <Partner>partner name</Partner>
    <Client>client name</Client>
    <Timestamp>2008-11-26T18:58:19.855</Timestamp>
    <Username> user</Username>
<Password>b578835bd9ae072013f7a9c8139cc7ef92326ac03d8d1ebf50ebda512f4dd
ec0</Password>
   </Header>
<RTMTriggerTemplate>
  <TriggerEmailTarget>
<Folder>folder name</Folder>
<Template> RTM Template Name</Template>
<Defer>
<MaximumDeferrableMinutes>2</MaximumDeferrableMinutes>
</Defer>
<ActionAfterDeferredTime>true</ActionAfterDeferredTime>
<ToEmailAddress>
  <EventEmailAddress>
    <EmailAddress>toemail@dom.com</EmailAddress>
    <FromEmailAddress>fromemail@dom.com</FromEmailAddress>
    <FromDisplayName>fromname</FromDisplayName>
    <ReplyToEmailAddress>replytoemail@dom.com</ReplyToEmailAddress>
    <ReplyToDisplayName>repltoname</ReplyToDisplayName>
    <EventVariables>
<Variable>
  <Name>var1</Name>
  <Value>varvalue1</Value>
</Variable>
<Variable>
  <Name>var2</Name>
  <Value>varvalue2</Value>
</Variable>
    </EventVariables>
    <ProfileAttributes>
<Attribute>
  <Key>attr1</Key>
  <Value>attrvalue1</Value>
</Attribute>
<Attribute>
  <Key>attr2</Key>
  <Value>attrvalue2</Value>
</Attribute>
    </ProfileAttributes>
```

```
    <TrackedUrls>
<URL label="my URL label">http://url.com</URL>
    </TrackedUrls>
  </EventEmailAddress>
</ToEmailAddress>
<ToEmailAddress>
<EventEmailAddress>
          <EmailProfile>
                    <EmailAddress>xzhu@epsilon.com</EmailAddress>
                     <ProfileKey>99</ProfileKey>
            </EmailProfile>
</EventEmailAddress>
</ToEmailAddress>
  </TriggerEmailTarget>
</RTMTriggerTemplate>
</rtmreq:RTMTriggerRequest>
```

RTMTrigger Responses

When you issue an RTM Simple Object Access Protocol (SOAP) API request, DREAM replies with a response object, called the RTMTriggerResponse. This section of the guide provides information on the RTMTriggerResponse, including:

- RTMTrigger Response elements
- RTMTrigger Response sample
- Available response codes

RTMTrigger Response elements

The following table lists the elements you can receive in the SOAP message in response to an RTMTriggerRequest message.

TABLE 9: RTMTriggerResponse elements

Element Name	Description	Nesting
`<rtmresp:TriggerResponse>`	Contains all the information associated with response information.	No nesting required; this is the root element.
`<rtmresp:Code>`	Code returned in response to the RTM request.	Nested within the TriggerResponse element. Mandatory.
`<rtmresp:Description>`	Description of the trigger request's response code.	Nested within the TriggerResponse element. Optional.
`<ResultData>`	Contains the elements that summarize the results of the RTM request.	Nested within the TriggerResponse element. Optional.
`<TransactionResult>`	Describes details of each submitted transaction.	Nested within the ResultData element. One per each submitted (successfully or otherwise) transaction. Mandatory.
`<Result>`	Contains a numeric result code for this transaction.	Nested within the TransactionResult. Mandatory.
`<TransactionID>`	Unique value assigned by the RTM to each successful trigger request destination.	Nested within the TransactionResult. Mandatory.
`<Description>`	Describes the result code of this transaction.	Nested within the TransactionResult. Mandatory.
`<TrackingID>`	Tracking ID assigned by the user to this transaction (if exists).	Nested within the TransactionResult. Optional.

TABLE 9: RTMTriggerResponse elements

Element Name	Description	Nesting
<ProfileKey>	This is a Profile Key for this transaction.	Nested within the TransactionResult. Mandatory.

RTMTrigger Response sample

The following is an example of a response to an RTM trigger request (without SOAP envelope).

```
<rtmresp:TriggerResponse xmlns:rtmresp="DREAMRTMResp.xsd" >
  <rtmresp:Code>-5</rtmresp:Code>
  <rtmresp:Description>Message failed to send. Please check the content
of the email for invalid scripting parameters</rtmresp:Description>
  <ResultData>
    <TransactionResult>
      <Result>1</Result>
      <Description>Transaction was processed successfully</Description>
      <TransactionID>1234567890</TransactionID>
      <TrackingID>trackid</TrackingID>
      <ProfileKey>user@some.com</ProfileKey>
    </TransactionResult>
  </ResultData>
</rtmresp:TriggerResponse>
```

Available response codes

Nested within the response object are two codes, the *RTM Response* code and the *Result* code, as seen in the following table:

TABLE 10: RTMTrigger Response Codes

Code	Description
RTMResp:Code	Provides information regarding the RTM message and whether it was successfully delivered or why it was unable to be delivered
Result	Provides the status of the RTM response transaction

The following tables provide the returnable values and descriptions for both codes.

TABLE 11: Potential values for <RTMResp:Code>

Code	Return Message
1	Your request was successfully received and processed
-2	No USERNAME field provided in the HTTP header

Code	Return Message
-3	No PASSWORD field provided in the HTTP header
-8	The request type supplied is not valid
-9	The request is too big (configurable size in configuration file)
-10	No PARTNER field provided in the HTTP header
-11	No CLIENT field provided in the HTTP header
-12	No TIMESTAMP field provided in the HTTP header
-13	The RTM request was rejected because the RTM template was not armed or the template does not exist
-20	The request is invalid (CONTENT_LENGTH HTTP header is missing or invalid)
-101	User authentication failed
-102	User does not exist or does not have RTM privileges
-200	XML doc parsing failed (mal-formed)
-280	The request XML is invalid according to its schema
-283	The maximum deferrable time is earlier than current time
-782	Internal error - please contact Support

TABLE 12: Potential values for `<Result>` code

Status	Description
0	Transaction submitted
1	Internal error submitting transaction

Epsilon recommends that you use only the response code as the basis for parsing the result of a request. Use of the descriptive text as a test condition is not recommended because the wording of the text may change in future releases.

REQUESTING AN RTM STATUS MESSAGE

The MakeGetStatusRequest API can be used to request a return response from DREAM regarding the status of a specific RTM message transaction for example, whether the message was successfully delivered to the recipient or failed.

This section discusses the following topics regarding the MakeGetStatusRequest API:

- Creating RTMGetStatusRequests
- RTMGetStatus Responses

Creating RTMGetStatusRequests

The MakeGetStatusRequest method is used to request the status of a specific RTM message and consists of two SOAP messages, the request and the response, as seen in the table below:.

TABLE 13: SOAP messages for MakeGetStatusRequest

Message	Description
RTMGetStatusRequest	Request sent to obtain the status of an RTM message Note: RTM status information is stored for 90 days. This means no data will be returned for a GetStatus request for a transaction ID over 90 days old.
RTMGetStatusResponse	Received as a response to the RTMGetStatusRequest message

This section outlines the use of the RTMGetStatusRequest and includes:

- RTMGetStatus Request elements
- RTMGetStatus Request sample code

RTMGetStatus Request elements

The following table lists the elements you can use in the SOAP message to query the status of RTM messages sent by a specific client on the basis of either the RTM Transaction ID or Tracking ID.

Note: The recommended method to obtain the status of an RTM request uses the Transaction ID. Usage of the Tracking ID is not recommended.

TABLE 14: RTMGetStatusRequest elements

Element Name	Description	Nesting
`<RTMGetStatusRequest>`	Root element. Contains all the information associated with finding RTM messages status.	No nesting required; this is the root element.
`<Header>`	Specifies validation and authentication parameters of this message.	Nested within the RTMGetStatusRequest element. Mandatory.

TABLE 14: RTMGetStatusRequest elements

Element Name	Description	Nesting
<RTMGetStatus>	Used to find the statuses of RTM messages sent by a client by Transaction ID or by Tracking ID. The maximum number of transactions you can query by Tracking ID varies depending on the number of internal events that are associated with each transaction.	Nested within the RTMGetStatusRequest element. Mandatory.
<Transactions>	Contains Transaction IDs or Tracking IDs of the message.	Nested within the RTMGetStatus element. Mandatory.
<IDs>	Contains an array of ID elements used to query statuses.	Nested within the Transactions element. Mandatory.
<Type>	Type of ID contained in ID element. Can be either TrackingID or TransactionID.	Nested within the Transactions element. The default value is TransactionID. Optional.
<ID>	Contains a single ID (transaction or tracking) to query.	Nested within the IDs element. Can be specified multiple times if status for more than a single ID is requested. Mandatory.

RTMGetStatus Request sample code

The following example has steps to query the status of multiple RTM transactions using the Tracking ID (SOAP envelop is omitted).

```
<rtmreq:RTMGetStatusRequest xmlns:rtmreq="DREAMRTMWebletReq.xsd">

<RTMGetStatus>
  <Transactions>
    <Type>TrackingID</Type>
    <IDs>
      <ID>1234567</ID>
      <ID>my Tracking ID</ID>
    </IDs>
  </Transactions>
</RTMGetStatus>
</rtmreq:RTMGetStatusRequest>
```

RTMGetStatus Responses

When an RTMGetStatusRequest call is issued, DREAM replies with a response object, called the RTMGetStatusResponse. This response contains information on the status of the RTM deployment.

This section of the guide provides information on the RTMGetStatusResponse, including:

- RTMGetStatus Response elements
- RTMGetStatus Response sample
- RTMGetStatus Response codes

RTMGetStatus Response elements

The following table lists the elements you can receive in the SOAP message in response to an RTMGetStatus Request.

TABLE 15: RTMGetStatusResponse elements

Element Name	Description	Nesting
`<rtmresp:GetStatusResponse>`	Contains all the information associated with response information.	No nesting required; this is the root element.
`<rtmresp:Code>`	Code returned in response to the RTM request.	Nested within the GetStatusResponse element. Mandatory.
`<rtmresp:Description>`	Description of the trigger request's response code.	Nested within the GetStatusResponse element. Optional.
`<ResultData>`	Contains the elements that summarize the results of the RTM request.	Nested within the GetStatusResponse element. Optional.
`<TransactionStatus>`	Contains status of transaction requested by user.	Nested in the ResultData element. One element per each transaction status requested. Mandatory.
`<TransactionID>`	Unique value assigned by the RTM to each successful trigger request destination.	Nested within the TransactionStatus element. Mandatory.
`<TrackingID>`	Tracking ID assigned by the user to this transaction (if exists).	Nested in the TransactionStatus element. Optional.
`<ForRTMEmailToEmailAddresses>`	Contains the transaction status, recipient's email address, and the number of times the request was sent and received.	Nested within the TransactionStatus element. Mandatory.

TABLE 15: RTMGetStatusResponse elements

Element Name	Description	Nesting
<ReceivedTime>	Date and time the RTM message was received (transaction created time).	Nested within the ForRTMEmailToEmailAddresses element. Mandatory.
<RecipientProcessingDetail>	Contains the recipient's email address, message status, and execution time.	Nested within the ForRTMEmailToEmailAddresses element. Mandatory.
<EmailAddress>	Recipient's email address.	Nested within the RecipientProcessingDetail element. Mandatory.
<Status>	Outcome of the RTM Transaction (status code).	Nested within the RecipientProcessingDetail element. Mandatory.
<Description>	Meaningful description of the transaction status.	Nested in the RecipientProcessingDetail element. Mandatory.
<ProcessedTime>	Last activity time for this transaction (if there was an activity).	Nested in the RecipientProcessingDetail element. Optional.

RTMGetStatus Response sample

The following example is a response to a status request of an RTM message (without SOAP envelope).

```
<rtmresp:GetStatusResponse xmlns:rtmresp="DREAMRTMResp.xsd">
  <rtmresp:Code>1</rtmresp:Code>
  <rtmresp:Description>Your request was successfully received and
processed.
  </rtmresp:Description>
  <ResultData>
    <TransactionStatus>
      <TransactionID>4504029124283345</TransactionID>
      <ForRTMEmailToEmailAddresses>
        <ReceivedTime>2004-06-02T04:32:11+04:00</ReceivedTime>
        <RecipientProcessingDetail>
          <EmailAddress>somebody@somecompany.com</EmailAddress>
          <Status>1</Status>
          <Description> Transaction was processed successfully
           </Description>
          <ProcessedTime>2004-06-02T04:36:42-04:00</ProcessedTime>
        </RecipientProcessingDetail>
      </ForRTMEmailToEmailAddresses>
    </TransactionStatus>
  </ResultData>
</rtmresp:GetStatusResponse>
```

RTMGetStatus Response codes

Nested within the response object are two codes, the *RTMGetStatus Response* code and the *Status* code, as seen in the following table. Each code includes a numeric success or failure indicator, along with a single line of text describing the code.

TABLE 16: RTMGetStatusResponse Code Types

Code	Description
RTMResp:Code	Provides additional information regarding why the RTM message was unable to be delivered
Status	Provides the status of the RTM message transaction

The following tables provide the values and descriptions for both codes:

TABLE 17: Potential codes for `<RTMResp:Code>`:

Code	Return Message
2	The requested transaction status(es) cannot be found.
1	Your request was successfully received and processed.
-2	No USERNAME field provided in the HTTP header
-3	No PASSWORD field provided in the HTTP header
-8	The request type supplied is not valid.
-9	The request is too big (configurable size in configuration file)
-10	No PARTNER field provided in the HTTP header
-11	No CLIENT field provided in the HTTP header
-12	No TIMESTAMP field provided in the HTTP header
-20	The request is invalid (CONTENT_LENGTH HTTP header is missing or invalid)
-101	User authentication failed.
-102	User does not exist or does not have RTM privileges.
-200	XML doc parsing failed (mal-formed).
-280	The request XML is invalid according to its schema.
-283	The maximum deferrable time is earlier than current time.
-782	Internal error. Contact Support.

TABLE 18: Potential codes for `<Status>`

Status	Description
2	Transaction message is being sent.
1	Transaction was processed successfully.
0	Unknown transaction or transaction status is not available yet
-1	Recipient email address was invalid as the syntax of email address is invalid
-5	Message failed to send. Please check the content of the email for invalid scripting parameters.
-6	Recipient's profile contains an invalid profile key.
-7	Recipient's email belongs to a wireless domain.
-8	Failed to process the transaction due to a problem connecting to the database. The message will not be retried.
-11	Transaction's message experienced a soft bounceback. The recipient may still receive the message.
-12	Transaction's message experienced a hard bounceback, often due to an invalid email address.
-21	Failed to process the transaction
-22	Transaction's processing was deferred
-24	Recipient email address is unsubscribed. No delivery was made to it.
-25	Recipient email address was invalid due to too many bounces.
-26	Transaction was rejected because the mailing in request had not been armed yet

Note: Status codes 0, -11 and -22 indicate the RTM request is still in process. Once the request has completed, these status codes will be updated.

Epsilon recommends that you use only the response code as the basis for parsing the result of a request. Use of the descriptive text as a test condition is not recommended because the wording of the text may change in future releases.

REPORTING ON REAL-TIME MESSAGES

You can generate RTM reports using the Reports module of the DREAM application.

This section includes the following topic:

- RTM Reports in DREAM

RTM Reports in DREAM

There are several DREAM reports you can use to analyze RTM-related data, including the following:

TABLE 19: RTM Reports

Report	Description
Summary Report	Provides an analysis and summary of the delivery and response data for RTM messages.
Activity Download Report	Allows you to download files of individual records associated with activity for a specific mailing or mailings, such as sent, replies, link clicks, opt-outs, and HTML opens. The data can be downloaded in an Excel or tab-delimited format, emailed, sent by FTP, or used to create a sub list. The RTM information captured in this report includes **RTM Transaction ID**, which is a unique identifier for the transaction.
Executive Report	Provides a high-level summary of the delivery and response data for RTM messages.
Weekly Status Report	Provides a weekly report of mailing activity for the selected campaign.

Real-time reporting functionality

Real-time reporting functionality includes:

- The Real-time report is available for all templates at the Mailing_ID level.
- Similar to the Executive Report, the data source for the Real Time report is the mailing summary table (summary_tbl). However, the Real Time report data is constantly being refreshed with the latest mailing activity data every 120 seconds. The Executive Report, although it reflects the latest mailing results, refreshes its data at a slower rate. The xxx and the xx report don't have the auto-refresh functionality. These reports gather data as a snapshot at the time of execution.
- The longer you wait to generate the report after sending a mailing, the higher the Click/Open counts will be.
- From the backend prospective, activity data — send, click, and open activities — is uploaded by the reporting system every 10,000 records or 60 seconds whichever comes first resulting in the activity data being uploaded all the time.
- The ability to change a URL link after performing a mailing.

Tracking URLs for Reports

Most URLs in an RTM message can be tracked even when the DREAM database is unavailable. If the URLs have been tracked in previous messages (and are therefore saved in DREAM), clicks to these URLs in the RTM message are tracked. However, if the URLs have not been previously tracked, they are not tracked when the DREAM database is unavailable for system maintenance and other similar reasons.

Note: Ensure that the RTM template URLs are saved in DREAM.

When the DREAM database is available again, DREAM reports on URL-tracking are updated in batch mode. As a result, reports containing URL data may not be immediately updated when DREAM functions.