HCLSoftware

Unica Deliver V12.1.6 Transactional Message Administration Guide

Contents

Chapter 1. About Deliver and transactional message.....1

Using Deliver to send transactional message.....1

How transactional message works in deliver.....2

Transactional messaging and standard messaging compared.....4

Message design for transactional message.....5

Recipient information for transactional message.....6

Global message suppression and transactional message.....7

Attachments to transactional email.....7

What digital marketers do for transactional message.....8

About enabling mailings for transactional message.....9

About responding to errors related to transactional email.....9

What application developers do for transactional message.....10

Choosing between REST API and SOAP API.....11

Chapter 2. Integrating with the Deliver transactional message service.....12

Identifying transaction events.....12

Connecting to recipient data.....12

Connecting to the Deliver TMS.....13

About Deliver TMS addresses.....13

The WSDL for Deliver transactional email.....14

About providing authentication credentials for transactional email.....14

About providing attachments in transactional email requests.....15

Constructing the transactional email request.....16

mailingCode.....16

audienceID.....17

Fields.....17

Cell codes.....18

additionalOptions.....18

attachments.....18

Tracking fields.....19

locale.....19

userName.....20

password.....20

Chapter 3. Deliver sample TMS client.....21

Chapter 4. The Deliver Transactional Mailing Service API.....22

NameValuePair.....22

Response.....22

AdvisoryMessage.....23

Error messages for transactional email.....24

Chapter 5. Sample client.....28

Chapter 6. About Dynamic Transactional Images.....31

Using dynamic transactional images in transactional email messages.....31

Defining an image label for dynamic transactional images.....31

How to specify dynamic transactional images in the transactional email request.....33

Chapter 1. About Deliver and transactional message

A transactional message is a single message sent in response to a specific, predetermined transaction detected in your business systems. Unica provides the Deliver Transactional messaging Service (TMS) as a hosted web service to process transactional messages. Digital marketers work with application developers to integrate corporate transaction management systems with the Deliver TMS.

Note: Deliver supports the following channels along with email. In this guide, the term message applies to all channels.

- SMS
- WhatsApp
- Push

Sending transactional message lets you respond automatically to specific customer or customer-related activities with a personalized message. Transactional messages tend to have higher open rates than other types of marketing message. Message recipients are more likely to open a message related to a transaction they recognize or expect than they are to open an unsolicited message.

You can use any event that you can detect in your business systems to prompt a transactional message. For example, you can send a transactional message when an individual subscribes to your monthly newsletter or requests information in response to an digital marketing campaign.

The following topics provide an overview of the Deliver TMS and the roles played by digital marketers and application developers when using transactional message.

- Using Deliver to send transactional message on page 1
- What digital marketers do for transactional messages on page 8
- What application developers do for transactional message on page 10
- Choosing between REST API and SOAP API on page 11

Using Deliver to send transactional message

Implementing transactional message requires collaboration between digital marketers and application developers. All parties must have a general understanding of the required systems and workflow. Everyone involved must be familiar with the various roles that each contributor plays in the transactional message implementation.

Using Deliver to send transactional message involves the following activities and systems.

Identify transactions that require an automated message response

The digital marketing team determines the types of transactions that require an automated message response.

Transactional message is based on message content and recipient information referenced in a standard Deliver messaging. You can enable any standard messaging for transactional message. For details, see About enabling messagings for transactional message on page 9.

Deliver Transactional messaging Service

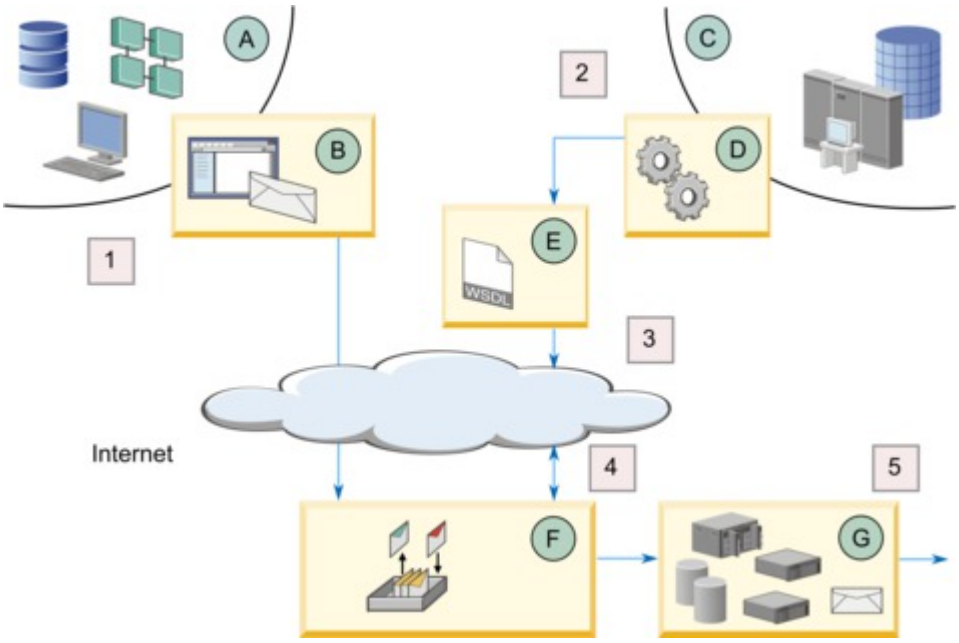
Unica hosts the Deliver Transactional messaging Service (TMS) as a web service to automatically process transactional message requests.

Develop systems to monitor transactions and request transactional message

Application developers must create a client application that receives transaction notifications from corporate business systems and submits web service requests to the Deliver TMS. Transactional email messages sent using Deliver can include attachments. The requests must provide personalization data and attachment content.

How transactional message works in Deliver

Digital marketers use Campaign and Deliver to configure messages and enable messagings for transactional message. Application developers create a transactional message client and integrate it with other business systems.



The following table describes the how locally installed HCL Unica systems and HCL Unica hosted services operate together to send transactional message.

	Element		Related activity
A	**Unica Campaign**	**1**	The digital marketing team enables a messaging (**B**) for transactional message. Any standard

	Element		Related activity
	Campaign provides interfaces to compose message, configure messagings, and enable messagings for transactional message.		messaging can be enabled for transactional message. The messaging configuration contains the mailing code that Deliver uses to identify the messaging. The mailing code is required for transactional message requests. Digital marketers must communicate the mailing code to application developers that create the local transactional message client (**E**).
B	**Deliver messaging** The messaging configuration references a message document that provides message content, including personalization fields that serve as placeholders for recipient data.		
C	**Your corporate business systems and databases** Various business systems and databases can provide the personalization data and attachment information that a transactional message requires.	**2**	A transaction occurs in your business systems (**C**). The transaction is considered a *transaction event* Digital marketers and application developers consult in advance to identify the personalization data and attachments required to respond to a transaction event. The transaction monitoring system (**D**) does the following. • Detects the transaction event • Recognizes that the event type has been designated as one that requires an message response • Provides the required personalization data and attachment content to the transactional email client (**E**)
D	**Your transaction monitoring systems** Corporate transactional applications monitor business systems to detect the types of transactions that trigger transactional message. You can use almost any business activity as a trigger for a transactional message. System administrators configure the monitoring systems to detect the specified transactions and provide message personalization information.		
E	**Local transactional message client** Client application that submits web service requests to the Deliver Transactional messaging Service (TMS). Unica provides a REST API documentation and a WSDL for use as a development guide to create the client. The client resides in your local network. It is not part of the Deliver TMS.	**3**	The TMS client provides access credentials for HCL Unica hosted services in a REST or a SOAP request over HTTPS to the Deliver TMS (**F**). The request specifies the Deliver mailing code and provides values for all personalization fields contained in the email. The request also includes attachments, as necessary.
F	**Deliver Transactional messaging Service (TMS)** A web service that resides in HCL Unica hosted services. The service receives and processes transactional email requests from the local transactional message client.	**4**	The Deliver TMS reviews the web service request. If the request provides all information required, the Deliver TMS forwards the request, including

	Element		Related activity
			attachments, to the Deliver mailing infrastructure (**G**) for transmission. If the TMS encounters a problem, it returns an error message to the local client application describing the issue.
G	**Deliver mailing infrastructure** Deliver components and servers maintained in Unica data centers that assemble and transmit the transactional message messages.	**5**	Deliver transmits and tracks a single personalized message addressed to the recipient identified in the message request. The local Campaign installation retrieves contact and response data and stores it in the Deliver system tables.

Transactional messaging and standard messaging compared

Deliver constructs and sends transactional messages differently than it does standard personalized message. During a standard messaging run, the system processes potentially large volumes of individually personalized messages. However, for transactional message, Deliver performs the same personalization operations for multiple web service requests, but each request processes only one message at a time.

You can enable any standard Deliver messaging for transactional message. Most messaging features available to standard messagings remain available when you enable the messaging for transactional message. Content elements available in standard message, such as personalization fields, text, images, HTML snippets, and hyperlinks are also available in transactional email message. However, some differences exist in the messaging features available in standard and transactional message.

The following table compares key features available in standard and transactional email message.

Feature	Standard messaging	Transactional messaging
Transmit HTML, HTML and text, and text-only versions.	X	X
Output List Table (OLT) to specify personalization data	X OLT contains all recipient data used to personalize email.	Not used. Can be used for testing.
OLT personalization fields	X	X
Built-in personalization fields	X	X
Constant personalization fields (constants)	X	X
Conditional content	X	X

Feature	Standard messaging	Transactional messaging
Advanced scripting for email	X	X
Message preview	X	X Preview not available for attachments.
Messaging results appear in standard Deliver performance reports	X	X
Track links in email messages	X	X
Tracking Audience ID as a contact attribute	X	X
Use personalization fields for contact tracking	X	X
Additional URL parameters for link tracking	X	X
Global email suppression	X	
Email attachments		X

Message design for transactional message

Every messaging that you enable for transactional message must reference a Deliver document. Any Deliver document created for use in a standard messaging can also be sent as a transactional message. The Deliver document defines the content of the message, including text, images (for email and Whatsapp messages), links, and conditional content. The message document also contains personalization fields that serve as placeholders for data specific to the recipient, including email addresses, phone numbers for SMS and Whatsapp.

As part of the transactional message request, the local transactional message client provides the unique mailing code that identifies the messaging. Deliver uses this code to determine which message document it must use to create the transactional message. The transactional message request must also provide the values required to populate the personalization fields in the Deliver document that the messaging references. If all of the required personalization values are not present in the request, Deliver will not process the transactional message and the system returns an error.

Digital marketers and application developers must review each message design in advance to ensure that the message request provides all of the information required. If the design of the document changes, for example, to change the personalization information in the message, the design of the transactional message request must be updated to reflect the change.

About identifying the sender of the transactional message

The message document used in a transactional email must contain a **From** address that is displayed to the recipient. If you specify the From address using a personalization field, the email domain of the address must match the email domain registered with Unica for your Deliver account.

Recipient information for transactional message

To personalize transactional message, you must provide recipient information in the web service request that you send to the Transactional messaging Service (TMS). Unlike other forms of personalized marketing message that you can send through HCL Unica, transactional messages do not retrieve recipient information from an Output List Table (OLT).

Digital marketers and application designers must ensure that the local transactional message client can provide values to populate all of the personalization fields that are used in the transactional message. Each web service request provides address and personalization information that is specific to the message recipient.

You can enable a existing standard messaging to send messages as transactional message. However, you do not need to reference an OLT to use a standard messaging to send transactional message. The recipient information required by the transactional message is contained in the web service request.

Personalization fields used in a transactional message

The local transactional message client identifies each personalization field as a separate name-value pair in the web service request that it submits to the hosted transactional messaging service. The client application must specify the name of each personalization field that is contained in the message. The client must also access the business systems and databases that provide the required personalization values.

The messaging that you enable for transactional message must reference a corresponding document. The document provides the structure and content of the transactional message. The document also includes the names of the personalization fields that are included in the message. The personalization fields are added to the document as placeholders that accept specific information about the recipient when the messaging service assembles and transmits the message.

The Summary Mailing tab provides a field that is labeled **Complete field list for this mailing** to identify the personalization fields that are included in the document that is referenced by the messaging. The web service request must contain information for each of the fields on the list. The names of the personalization fields in the web service request must exactly match the names that appear in the document.

The web service request must also provide the data that is required to complete the message, including values for each personalization field that is included in the message. The digital marketing team must consult with application developers to identify and locate all of the information that the transactional message client must provide.

The transactional messaging service evaluates each web service request to determine if the request provides all of the name-value pairs that are required by the transactional message. The request fails if the personalization field names, values, or data types do not match the requirements for the message.

Personalization fields for additional link or contact tracking in transactional message

If you request that Unica perform additional link or contact tracking, every transactional message request must include the name and value of personalization fields that are used for additional tracking.

The web service request for transactional message must include parameters that specify the name and value of the tracking personalization fields. The digital marketing team must provide application developers with the following information.

- Names of the tracking personalization fields
- Expected personalization field values and data types
- Format or length restrictions

Deliver does not validate the uniqueness of the fields you specify for additional tracking. To distinguish additional tracking data for transactional message from data that is collected for standard message, establish internal procedures or naming conventions to ensure unique personalization field names. Avoid specifying the same personalization field name for link or contact tracking in standard messagings and transactional messagings.

Global message suppression and transactional message

Unica Deliver does not apply global email suppressions to transactional message requests.

To avoid violating laws regarding unwanted email delivery, such as CAN-SPAM, you must make your transactional systems aware of email addresses that must not receive email. Preventing transmission of transactional email to incorrect or unsubscribed addresses can also avoid deliverability problems caused by recipients that mark the transactional email as unwanted email.

Attachments to transactional email

The Deliver TMS SOAP service supports attaching files to transactional email messages. TMS REST API does not support attachments to be transmitted as part of request.

By attaching files to transactional email, you can provide the email recipient with extra personalized information. For example, you can send a transactional email to confirm a concert ticket purchase and use attachments to provide a printable ticket and seating map. You cannot specify attachments for standard mailings.

When you enable a mailing for transactional email, you must specify how many attachments you want to send with the email message. Deliver places limits on the size of individual attachments and the total size of all attachments.

The transactional email request that is submitted to the Deliver TMS must contain the document content and information about each attached document. The email marketing team must work with application developers to provide the following information about each attachment.

- The file name of the attachment
- MIME content type of the file
- Contents of the file

The method that is used to include attachments depends on the programming language and development tools that application developers use. For more information about how to provide attached content, see Attachments in transactional email requests on page 15.

Virus scanning for attachments

The Deliver Transactional Mailing Service does not scan attachments for computer viruses or other types of malware.

You are responsible for ensuring that none of the files that you attach to transactional email contain malicious code.

What digital marketers do for transactional messages

Using Deliver to send transactional message requires advance preparation and coordination between the digital marketing team and application developers responsible for your corporate transactional systems.

The following table lists typical activities that a digital marketer performs to prepare a messaging for transactional message.

Activity	Description
Identify the type of transaction that will initiate a transactional message request.	Confirm with the application development team that transaction systems can detect the transaction event that must trigger the email.
Compose and publish the Deliver document that you want to use for the transactional messaging.	Compose a Deliver document for transactional message in the same way as you would any other Deliver document.
Fully configure the messaging that you plan to enable for transactional email. In the configuration, indicate if transactional email includes attachments.	Reference the document that you have created for the transactional email. If you are including attachments, specify the number of attachments.
Identify all personalization fields used in the document. The mailing tab includes a list of personalization fields used in the document.	Application developers need to configure the transactional message client application to provide personalization information.
Provide the names and personalization field definitions to the application developers.	Each transactional message request web service request must specify the personalization field names and values as name-value pairs and specify the required data type.
Provide application developers with the mailing code for the messaging that you intend to enable for transactional message.	Developers require this information to identify the messaging for the Deliver TMS.
Determine if the transactional email includes attached files. If you use dynamic transactional images, add placeholders for the attachments in the email document that defines the transactional email message.	Consult with application developers regarding the number and size of the attached files. Agree on the labels to be used to identify dynamic transactional images.
Enable the messaging for transactional message, using links on the mailing tab.	The Deliver TMS begins accepting message requests immediately after you enable the messaging for transactional message.
Confirm that developers have finished configuring the local transactional message client before you enable the messaging for transactional message.	

About enabling messagings for transactional message

Transactional messages are based on standard messagings that have been enabled for transactional message. You enable a messaging for transactional message in Campaign, on the Summary Mailing tab. Consult the **Deliver Mailings** page to see which messagings are enabled for transactional message.

You can enable any Deliver messaging for transactional message. The transactional message request fails if you have not updated the messaging configuration to enable the messaging for transactional message. However, even after you enable a messaging to send individual messages as transactional message, you can run the same messaging as a standard messaging to perform a messaging campaign involving a large volume of messages.

Each transactional message request must include the mailing code that identifies the messaging. When you enable a messaging for transactional message, note the mailing code and provide it to application developers responsible for configuring the local transactional message client.

If you are attaching files to the transactional email messages, the messaging configuration must specify the number of attachments. Every transactional email message receives the number of attachments you specify. The number of attachments entered in the messaging configuration must match the number configured in the web service request submitted to the Deliver TMS. The attached files are sent only with transactional email. Deliver does not support sending attachments when you run a standard messaging, even if the messaging is also enabled for transactional email.

You can disable the messaging for transactional message at any time. For example, you must disable a messaging for transactional message to change the messaging configuration. The Deliver TMS does not accept transactional message requests while the messaging is disabled for transactional message.

As a best practice, before you enable a messaging for transactional message, fully test the messaging and preview the Deliver document the messaging references. Make certain that the messaging and the message meet your expectations and business objectives.

About editing messagings that are enabled for transactional message

To edit a messaging that is enabled for transactional message, you must disable the messaging for transactional message before you begin.

After you finishing editing the messaging, you must re-enable the messaging for transactional message. During this process, the Deliver TMS does not respond to transactional requests for the disabled messaging. The local systems that monitor transaction events must be designed to temporarily store the message requests until you re-enable the messaging for transactional message.

About responding to errors related to transactional email

Sometimes a transactional email message does not transmit as expected. The failure could be due to problems with the message configuration or temporary problems with mailing resources. If the Deliver TMS determines that a problem exists, the web service returns an advisory error code to the local transactional email client.

The local transactional email client is responsible for error handling. Application developers must design the client to recognize the error messages that the Deliver TMS might return. For a list of error codes in the Deliver TMS, see Error messages for transactional email on page 24.

All parties must be prepared to respond to unforeseen email problems. If the problems relate to mailing configuration or message design, application developers might call on the digital marketing team to resolve the issue.

What application developers do for transactional message

Application developers create the local transactional message client that submits transactional message requests to the Unica Deliver Transactional messaging Service (TMS). The local transactional message client application processes each transactional message message by making a REST or a SOAP request to the Deliver TMS over an HTTPS connection.

Working with Deliver TMS using REST APIs

Unica provides REST API documentation for Transactional messaging service under **Settings > Messaging Settings** section of Unica application. It explains how API can be invoked, how user can authenticate for calling this API, what input headers and request payload need to be provided to call this API and what response is returned by this service.

Payload for REST API is very similar to the one for SOAP service, so you can relate closely request object need to be formed for REST service with details provided in this documentation on how to prepare request object for SOAP client.

Also, if you already have coded SOAP client for interacting with Deliver TMS, migrating to REST will not be a very difficult task.

Working with Deliver TMS using SOAP APIs

Unica provides a WSDL to allow developers to design the web service request. For more information about the WSDL, see WSDL for Deliver transactional email on page 14.

Creating the local transactional message client application requires coordination between application development team responsible for corporate transaction systems and the digital marketing team. Digital marketers provide information about the transactional message messages and attachments that are to be sent as transactional message.

The following table lists the activities that application developers must perform.

Activity	Description
Build the transactional monitoring systems required to detect the types of transaction events that trigger transactional message.	Consult with the Digital marketing team to determine which business or customer activities qualify as suitable transaction events.
Code or configure a local transactional message client application to connect to the Deliver TMS and submit web service requests in response to specified transaction events. The client must be able to provide access credentials for your hosted message account. Structure web service requests to provide personalization information as name-value pairs.	Review Transactional Message service API documentation in **Settings** -> **Messaging Settings** section of Unica application if you plan to use REST APIs for transactional messages. Review the WSDL and sample transactional message client examples provided byUnica for guidance.

Activity	Description
The request must also specify the data type for each personalization field. Consult with the digital marketing team to determine if sending email attachments is a requirement.	
Coordinate with the digital marketing team to identify the sources for the personalization required by the transactional email messages and attachments.	The transactional message client must be able to connect to the business systems and databases that provide personalization values in the transactional message request.
Design and code error handling.	Code the client to handle errors reported by the Deliver TMS. Include features to notify administrators if the client cannot connect or if web service requests fail. For information about error codes returned by the Deliver TMS, see AdvisoryMessage on page 23.
Coordinate with the digital marketing team to test the integration of the local transactional message client with the Deliver TMS.	Digital marketers need to enable messagings for transactional message testing and specify test addresses. Application developers need to devise ways to simulate the target transactional events and provide expected transaction and personalization information to the local transactional message client.

Choosing between REST API and SOAP API

Unica provides both options of interacting with TMS service using REST or SOAP API.

SOAP API is legacy and traditional mechanism in which developers can use WSDL to build a client and invoke services as explained in rest of the document. It also allows you to provide attachments to sent via email so SOAP can be a preferred mechanism if your communication involves attachments like a ticket for a concert. However, SOAP API can send message only to a single user in an invocation.

REST API is a newer easy to use industry standard mechanism of exposing and consuming APIs based on HTTP protocol. It also provides an easy to follow documentation which explains API signature, request and header parameters, request payload and response details in a simple to understand JSON format. Deliver TMS REST API also allow you to send messages to multiple (upto 1000) users in a single invocation. However, it doesn't support attachments as of now.

Chapter 2. Integration with the Deliver transactional message service

The Deliver Transactional messaging System (TMS) is a web service hosted as part of HCL Unica hosted services. Application developers must work with digital marketers in your organization to create the client applications required to integrate your corporate transactional systems with the Deliver TMS. These client applications initiate each transactional message by making a SOAP request to the Deliver TMS over an HTTPS connection.

Client applications that submit the transactional message request must be able to perform the following actions.

- Identify transactions that trigger transactional message.
- Connect to marketing databases that contain data used to personalize the message
- Initiate® the transactional message request as a SOAP request
- Provide credentials to access the Deliver TMS over a secure connection
- Provide message information, including attachment contents
- Handle error messages returned by the Deliver TMS

All of the actions required for transactional message must occur without manual intervention. Unica provides a WSDL to assist developers that design the automated web service calls to the Deliver TMS. For more information, see the WSDL for Deliver transactional email on page 14 topic.

Identification of transaction events

Application developers must design the local transactional message client to interact with transaction monitoring systems used by your organization. The client must take transaction notifications as input and distinguish the type of transaction event that the transaction management system is reporting.

A transaction event can be any action that your transaction management systems can detect. For example, a transaction event might be a purchase, a customer request for service, an information request, or a change in customer account status.

Typically, your marketing organization determines the types of transactional events that warrant a transactional message request and the content of the message sent in response. Each transactional event requires a separate transactional message request to the Deliver TMS.

Access to recipient data

To personalize transactional messages, the local transactional message client must access business systems and databases that can provide recipient-specific data used in transactional message.

Application developers must collaborate with the digital marketing team to identify the personalization fields used in the message referenced by mailings enabled for transactional message. The summary Deliver mailing tab provides a link to a list of the personalization fields used in the message.

The personalization fields are placeholders for information specific to the recipient of the message. This information is provided by business systems and databases managed by your organization. The local transactional message client must be able to access these systems and retrieve data used as personalization field values.

Connection to the Deliver TMS

You must design the local transactional message client to automatically connect to the Unica data center assigned to your hosted message account. The client must be able to provide the appropriate authentication credentials as part of the web service request.

For information about which data center to specify, see Deliver TMS addresses on page 13.

For information about how to provide authentication credentials, see About providing authentication credentials for transactional message on page 14.

Deliver TMS addresses

Unica has established message domains reserved for processing transactional messages. You must configure your transactional message applications to request access to HCL Unica using the correct domain. The domain you use depends on which Unica data center processes your request.

To process transactional email through the Unica data center for North American, configure a connection to `tms-us.unicadeliver.com`.

To process transactional email through the Unica data center for the Europe, configure a connection to `tms-eu.unicadeliver.com`.

To process transactional email through the Unica data center for the India, configure a connection to `tms-in.unicadeliver.com`.

If you do not know which data center Unica uses to process your transactional email, contact technical support.

URL for TMS REST API service is as follows:

```
<URL for connecting to your assigned TMS>/delivertms/api/deliver/rest/v1/tms
```

We have introduced a new API that sends the message sequence number (envelope ID) in the response so that the mapping between audience ID and envelope ID remains with Journey. Journey can use the same mapping to process WhatsApp realtime replies that were getting delayed due to contact event arriving from RCT file processing, which took a lot of time.

The URL of the new API is as follows:

```
<URL for connecting to your assigned TMS>/delivertms/api/deliver/rest/v2/tms
```

The request body remains the same and as the previous API but the response format is updated to the following example:

```
{
 "[234]":
 [
   "SUCCESS",
   "100402000000012019"
 ],
  "[123, JohnDoe]":
 [
   "SUCCESS",
   "100402000000012018"
   ]
}
```

WSDL for Deliver transactional email

Unica provides two versions of the WSDL for building applications that access the Deliver TMS. The choice of WSDL depends on if you plan to use email attachments and on how the client application provides access credentia. WSDL describes how to construct an interface to the Deliver TMS that supports providing email attachments. It also describes how to specify access credentials as parameters in the web service request, instead of modifying the SOAP header.

The WSDL is available for download from the Unica data center that Unica has instructed you to use for connecting to the Deliver TMS. Construct the download URL as follows:

```
<URL for connecting to your assigned TMS>/delivertms/services/TMS?wsdl
```

Unica provides an example of how to create the local transactional email client based on the second-generation WSDL. To view the example, see Sample client on page 28

Note: If you have already constructed a transactional email client based on the first-generation WSDL you must recompile the code if you change to a client design based on the second-generation WSDL.

Second-generation WSDL

The second-generation WSDL describes how to construct an interface to the Deliver TMS that supports providing email attachments. This WSDL also describes how to specify access credentials as parameters in the web service request, instead of modifying the SOAP header.

The second-generation WSDL is available for download from the Unica data center that Unica has instructed you to use for connecting to the Deliver TMS. Construct the download URL as follows:

```
<URL for connecting to your assigned TMS>/delivertms/services/TMS?wsdl
```

Unica provides an example of how to create the local transactional email client based on the second-generation WSDL. To view the example, see Sample client on page 28

Note: If you have already constructed a transactional email client based on the first-generation WSDL you must recompile the code if you change to a client design based on the second-generation WSDL.

About providing authentication credentials for transactional message

The local transactional message client must be able to provide the authentication credentials required to connect to the Deliver TMS automatically.

When requesting transactional message, transactional applications must provide authentication credentials to access HCL Unica hosted services. You can provide the access credentials either by modifying the SOAP header or by providing them as parameters in the SOAP web service request.

The user name and password that the client must provide in the SOAP request are the user name and password assigned to your Deliver account. If you do not know these credentials, contact individuals in your organization who are responsible for maintaining your account, or contact Unica technical support.

Adding access credentials to the SOAP header

The Deliver TMS requires a user name and password in the web services request. Modifying the SOAP header in the web services request to include access credentials for HCL Unica hosted services is one method to access the Deliver TMS. You cannot use this method if you plan to use attachments with transactional email messages.

Before you begin

Before you begin, obtain the username and password that has been created for your Unica Deliver hosted email account.

About this task

The user name and password that you add to the SOAP header must be the username and password created for your Unica Deliver hosted email account, as shown below. The following code examples are based on an Axis2 SOAP library.

```
String userName = "<user name for your Unica Deliver account>";
String password = "<password for your Unica Deliver account>";
```

Modify the headers of your client application, as follows.

```
ServiceClient serviceClient = stub._getServiceClient();
        serviceClient.addStringHeader(new QName
("http://soap.tms.webservices.deliver.unica.com",
"userName", "ns2"), userName);
        serviceClient.addStringHeader(new QName
("http://soap.tms.webservices.deliver.unica.com",
"password", "ns2"), password);
```

Results

The modified headers must appear as shown in this example, where `UserName` and `Password` are the user name and password for your Deliver account.

```
<ns2:userName xmlns:ns2="http://soap.tms.webservices.deliver.unica.com">
UserName</ns2:userName>
<ns2:password xmlns:ns2="http://soap.tms.webservices.deliver.unica.com">
Password</ns2:password>
```

Access credentials as parameters

You can create a local transactional email client that accesses HCL Unica hosted services by submitting credentials as parameters in the web service request.

To submit access credentials as parameters, base the client design on the second-generation WSDL for the Deliver TMS. For more information about this WSDL, see Second-generation WSDL on page 14.

Attachments in transactional email requests

The Deliver TMS supports attaching one or more files to a transactional email message. The web services request must contain the file contents and information that describes each file.

The second-generation WSDL available from Unica describes how to configure a transactional email request that includes email attachments. The transactional email request must specify the following.

- Name of the file to be attached
- The MIME content type of the file
- Contents of the file

You pass attachments to the Deliver TMS inline as separate MIME parts using techniques described in either of the following standards recognized by the World Wide Web Consortium (W3C):

- SOAP Message Transmission Optimization Mechanism (MTOM)
- SOAP Messages with Attachments (SWA)

Consult the W3C web site for details about these standards.

Construction of the transactional email request

To access the Deliver TMS, the transactional email client must be able to provide credentials to connect to the TMS. It must also provide mailing and message information as part of the web service request. Unica provides two versions of a WSDL as a guide to structuring a SOAP request that includes all information required to send a transactional email.

For more information about the WSDL, see WSDL for Deliver transactional email on page 14.

The following topics describe the parameters that the client application must define in the web services request.

- mailingCode on page 16
- audienceID on page 17
- Fields on page 17
- Cell codes on page 18
- additionalOptions on page 18
- attachments on page 18
- Tracking fields on page 19
- locale on page 19
- userName on page 20
- password on page 20

mailingCode

The **mailingCode** parameter specifies the unique mailing code defined in the configuration of the Deliver mailing that you have enabled for transactional email. The **mailingCode** specifies the unique mailing code that is defined in the transactional mailing in Marketing Center. Marketing Center assigns the code to the mailing after the mailing is deployed. The code is displayed on the **Deploy** tab of the mailing.

Parameter name

mailingCode

Data type

String

Because the mailing code is unique within your account, you can use this parameter to identify the mailing. See the mailing for the specific value.

You can make the mailing code configurable in your local transactional email client application so that you can point to a different mailing if necessary.

audienceID

You define the **audienceID** parameter to correlate transactional email messages with email recipients when you perform additional reporting and processing.

Parameter name

audienceID

Data type

Varies.

You can identify one or more audience IDs for the transactional email recipient. Pass data for **audienceID** as an array of name-value pairs.

You can define any value or set of values for each **audienceID**. For example, you might use a promotional code, account type, geographical identifier, or all three as the value for **audienceID**.

Because each request sends transactional email to one individual, you can identify a specific email recipient by defining a unique **audienceID**. In this scenario, you might use a customer account number as a value for **audienceID**.

Deliver adds the name and value that you provide for `audienceID` to the tracking records in the Deliver system tables. Single audience ID values are stored in the **UCC_Envelope** table. Multiple audience ID values are stored in the **UCC_EnvelopeAttr** table.

In Unica Campaign, marketers can define various audience levels to identify individuals for tracking purposes. The Deliver TMS does not validate that the audience names that you provide for `audienceID` in a transactional email request match the audience names defined in Campaign. To avoid possible confusion when interpreting tracking results, consult with your marketing team to determine the correct audience names and values to pass.

Fields

Use the `fields` parameter to provide recipient-specific information to populate personalization fields defined in the transactional email message.

Parameter name

`fields`

Data type

Varies

Provide data as separate name-value pairs for each personalization field used in the email message.

You must identify each personalization field contained in the document referenced by the Deliver mailing. The name-value pair must provide the personalization field name as it is defined in the document. You can assign any value of the appropriate data type.

Cell codes

The `cellCodes` parameter is optional. By default, the Deliver TMS expects to receive a null value for this parameter. If you provide a value for `cellCodes`, you can pass only one cell code in each web service request.

Parameter name

`cellCodes`

Data type

String

You define cells and cell codes in Unica Campaign. A cell is a list of identifiers (such as customer or prospect IDs from your database). Each cell generated in a flowchart has a system-generated cell code. You can include a cell code in a web services request if you want to specify the cell that includes the message recipient.

For more information about defining cells, see the section on managing cells in the *Unica Campaign User's Guide.*

additionalOptions

This parameter is reserved for future use.

Parameter name

additionalOptions

Data type

Null

attachments

Use this parameter to specify files that are attached to the transactional email message. The web service request includes the content of the attached file in the format that you specify.

You can provide values for multiple elements in this parameter. If the email message includes a dynamic transactional image, you specify the image label as an element in this parameter.

Parameter name

attachments

Data Type

fileName: Name of the attached file. Pass the name as a string.

label: Used to identify dynamic transactional images. If the email includes dynamic transactional images, pass the label as a string. Otherwise, pass this value as null.

label: Used to identify personalized images. If the email includes a personalized image, pass the label as a string. You can omit the label if you are not using personalized images.

fileContent: the MIME content type of the attached file. The type is always base64Binary.

For more information about using the **label** attribute to identify dynamic transactional images, see About Dynamic Transactional Images on page 31.

Tracking fields

Use this parameter to specify personalized fields used for additional contact tracking.

Parameter name

`trackingFields`

Data type

Varies. Pass this data as separate name-value pairs for each personalization field used for tracking.

For more information about the contact and response information provided by additional contact tracking and the preparations required, see the section on additional tracking in the *Unica Deliver User's Guide*.

locale

This parameter specifies the locale, including the associated language, which is used for messages that are sent back from the Deliver Transactional Mailing Service.

This section lists the valid arguments for the supported locales.

Note: The list of locales that are available for the Deliver Transactional Mailing Service is different from the list of locales that area available for Marketing Center.

Parameter name

locale

Data type

Brazilian Portuguese: `arg.locale=pt_BR`

English: `arg.locale=en_US`

French: `arg.locale=fr`

German: `arg.locale=de`

Italian: `arg.locale=it`

Japanese: `arg.locale=ja`

Korean: `arg.locale=ko`

Russian: `arg.locale=ru`

Simplified Chinese: `arg.locale=zh_CN`

Spanish: `arg.locale=es`

userName

This parameter specifies the user name that is associated with your hosted email account. Specifying the user name is part of the authentication that is required to establish a connection with the Deliver TMS. You must also specify the password that is assigned to the hosted email account.

Parameter name

userName

Data type

String

For more information about your hosted email account and establishing a secure connection to HCL Unica hosted services, see the *Unica Deliver Startup and Administrator's Guide*.

password

This parameter specifies the password that is associated with your hosted email account. Specifying the password is part of the authentication that is required to establish a connection with the Deliver TMS. You must also specify the user name that is assigned to the hosted email account.

Parameter name

password

Data type

String

For more information about your hosted email account and establishing a secure connection to HCL Unica hosted services, see the *Unica Deliver Startup and Administrator's Guide*.

Chapter 3. Deliver sample TMS client

You can use Deliver sample TMS client to test connection to Unica hosted TMS service and send ad-hoc transactional mailings and send transactional messages using mailings configured to run as transactional using the following steps:

1. Navigate to `DELIVER_HOME/examples/tmsclient/bin`.
2. Open simpletest.properties in text editor and set below properties.
3. Set tms_url as per Deliver datacenter you are connecting to:

   US: https://tms-us.unicadeliver.com/delivertms/services/TMS

   EU: https://tms-eu.unicadeliver.com/delivertms/services/TMS

   IN: https://tms-in.unicadeliver.com/delivertms/services/TMS
4. Set arg.user and arg.pw to Deliver hosted account credentials (user configured in UNICA_HOSTED_SERVICES datasource). Password in arg.pw must be encrypted using DELIVER_HOME/bin/tmsPasswordEncrypt.bat/sh utility as per your OS platform.
5. Configure rest of the values as per the mailing you want to run (make sure mailing is enabled for TMS).
6. Run TMS test client using the following command as per your OS platform:

   ```
   runTMSTestClient.bat/sh simpletest.properties
   ```

Chapter 4. The Deliver Transactional Mailing Service API

The Deliver Transactional Mailing Service is a web service hosted in HCL Unica. It provides an API that contains one method called`sendMailing`.

The `sendMailing` method uses the following custom types.

- `NameValuePair`

  Provides methods to store mailing inputs as a name and corresponding value. See NameValuePair on page 22
- `Response`

  Provides status messages for the transactional email request. See Response on page 22
- `AdvisoryMessage`

  Provides detailed responses if the request status indicates a warning or error. See AdvisoryMessage on page 23

NameValuePair

The `NameValuePair` type provides the following methods for submitting parameter names.

You can submit the parameters as `string`, `numeric`, or `datetime` values.

Note: Although `NameValuePair` supports `datetime` parameter values, Deliver currently does not support using `datetime` values.

Method	Parameter
`setName`	`name`
`setValueAsString`	`valueAsString`
`setValueAsNumeric`	`valueAsNumeric`
`setValueAsDate`	`valueAsDate`
`setValueDataType`	`valueDataType`

Response

The **Response** custom type provides general acknowledgment messages that indicate if the email request was successful, or if the request resulted in an error or warning.

The following table lists the status types and related codes for **Response**. These codes are considered high-level status codes that describe the success or failure of the email request. The **AdvisoryMessage** custom type provides access to more detailed messages that describe the reasons for a failed email request.

Status Type	Description	Code
`STATUS_SUCCESS`	The request to the Deliver TMS is successful.	0

Status Type	Description	Code
`STATUS_WARNING`	The request encountered at least one warning, but no errors. The client can query the `AdvisoryMessage` type for more detail.	1
`STATUS_ERROR`	The request encountered at least one error.	2

Methods for Response

The **Response** type contains the following methods.

Method	Description	Returns
`getAdvisoryMessages`	Returns the list of advisory messages.	`AdvisoryMessage[]`
`getApiVersion`	Returns the API version.	String
`getStatusCode`	Returns the most severe advisory message in the response. For example, if the response contains a warning (code: 1) and an error (code: 2) then this method returns a 2.	Int

AdvisoryMessage

The **AdvisoryMessage** custom type provides more detail about status messages.

The following table lists the status types and related codes for the **AdvisoryMessage** type.

Status Type	Description	Code
`STATUS_LEVEL_INFO`	Informational message only. Does not reflect a failure in the call to the Deliver TMS.	0
`STATUS_LEVEL_WARNING`	The call to the Deliver TMS succeeded, but an issue exists that requires further investigation.	1
`STATUS_LEVEL_ERROR`	The call to the Deliver TMS failed.	2

Methods for AdvisoryMessage

The **AdvisoryMessage** type contains the following methods.

Method	Description	Returns
`getStatusLevel`	Returns the severity level of the status messages: Informational, Warning, or Error. For example, for `STATUS_LEVEL_ERROR`, this method returns the code: `2`.	Int

Method	Description	Returns
`getMessageCode`	Returns the code for an error message. For example, for the error, `INVALID_LOGIN`, this method returns the code: `1`.	Int
`getMessage`	Returns an error message.	String
`getDetailMessage`	Returns more details about an error message, if available.	String

Error messages for transactional email

The Deliver Transactional Mailing Service returns error messages and related codes.

The error messages that are described in the following table apply only to transactional email and transactional email requests.

Message	Description	Code
INVALID_LOGIN	The authentication credentials (user name, password, or both) provided during the call to the Deliver TMS do not match the credentials on file with Unica for your account. Review the **userName** and **password** parameters to ensure that the credentials have been specified correctly.	1
UNRECOGNIZED_MAILING_CODE	The mailing identified by the mailing code included in the call to the Deliver TMS is not enabled for transactional email. Check the mailing configuration to verify the code. Review the **mailingCode** parameter to ensure that the code has been configured correctly.The mailing code included in the call to the TMS does not match a transactional mailing. Check the mailing to verify the code. Review the **mailingCode** parameter to ensure that the code is correct.	2

Message	Description	Code
RUNTIME_EXCEPTION_ENCOUNTERED	The email request encountered an unexpected runtime exception. Contact Unica support.	3
ENVIRONMENT_EXCEPTION_ENCOUNTERED	The email request encountered an unexpected environment exception. Contact Unica support.	4
SMTP_CONNECTION_FAILURE	The connection to the SMTP server failed. Contact Unica support.	5
DOCUMENT_NOT_DEFINED_FOR_SPECIFIED_MAILING	The mailing enabled for transactional email does not reference an Deliver document. Examine the mailing configuration. The mailing configuration must reference an email document that defines the contents of the email message.The mailing does not include an email.	6
EMAIL_FAILED_TO_SEND	The email message was not transmitted successfully. You can try again or contact Unica support.	7
REQUIRED_PFS_MISSING	The email request did not specify names and values for all of the personalization fields required by the mailing. Confirm that the web service request defines all personalization fields used in the email document. The mailing configuration contains a link that lists all of the required personalization fields.The email request did not specify names and values for all of the fields that are required by the mailing. Confirm that the web service request defines all fields that are used in the email. The mailing lists all of the required fields.	8

Message	Description	Code
AUDIENCE_ID_MISSING	The request does not include an audience ID. Confirm that the web service request defines a value for the **audienceID** parameter.	9
ATTACHMENT_NUMBER_MISMATCH	The number of attachments defined in the mailing does not match the number passed in the web service request. Review the mailing configuration and the web service request. The mailing configuration and web service request must specify the same number of attachments. The number of attachments that is defined in the mailing does not match the number that is passed in the web service request. Review the mailing and the web service request. The mailing and web service request must specify the same number of attachments.	12
ATTACHMENT_SIZE_EXCEEDED	The size of one of the attachments exceeded the maximum allowed attachment size. Individual attachments cannot exceed 1 MB.	13
TOTAL_ATTACHMENT_SIZE_PER_MESSAGE_EXCEEDED	The total size of all attachments in the request exceeded the maximum total attachment size per message. The total size of all attachments cannot exceed 2 MB.	14
TMS_MAILING_ATTACHMENTS_LABEL_NOT_FOUND	A label provided in the **attachments** parameter (typically to identify a dynamic transactional image) is missing or does not match the label contained in the email document. NOTE: Attachment labels are case-sensitive; labels entered in the web service request must exactly match the label as entered	15

Message	Description	Code
	in the mail document. A label that is provided in the **attachments** parameter (typically to identify a personalized image) is missing or does not match the label in the email. NOTE: Attachment labels are case-sensitive; labels in the web service request must exactly match the label in the email.	
TMS_MAILING_ATTACHMENTS_LABEL_DUPLICATED	A label provided in the **attachments** parameter (typically to identify a dynamic transactional image) appears multiple times in the web service request. In the web service request, attachment labels must be unique. A label that is provided in the **attachments** parameter (typically to identify a personalized image) occurs multiple times in the web service request. In the web service request, attachment labels must be unique.	16

Chapter 5. Sample client

Unica provides sample transactional message clients to guide application developers that create client applications that make web service calls to the Deliver Transactional Messaging Service (TMS). This sample client is based on the second-generation WSDL.

The major difference between this WSDL and the earlier version is that this version supports using attachments with transactional message and provides authentication credentials as parameters.

The following program is an example that demonstrates how to structure a request to the Deliver Transactional Mailing Service. Review this example as an illustration of how to work with the application programming interface for the Deliver TMS.

This example is based on libraries for axis2 1.3. For details about axis2 1.3, see the following website: http://ws.apache.org/axis2/ .

```
public class SampleTestClient {
 public static void main(String[] args) throws AxisFault, RemoteException {
  /**
         * The sendMailing method of the TMS webservice requires:
         * 1) proper authentication information - a valid username and password recognized by the TMS
         * 2) a mailingCode to identify the Mailing that contains the document to be sent
         * 3) an audience identifier (used primarily for tracking)
         * 4) personalized fields that will be merged into the document to be sent
         * 5) optional cellCode(s) associated to the audience identifier.
         * 6) optional additionalOptions - there are currently no additional options supported,
         *      but is here for future use.  For now this parameter can be left as null.
         * 7) optional locale for the response messages to be returned otherwise default is "en" (Locale.US)
 for english
   */

  // authentication information
  String userName = "MyTMSUserName";
  String password = "MyTMSPassword";

  // mailing code
        String mailingCode = "mailing 123";

        // audience id: note, an audience id is comprised of at least one name value pair.
  // a custom type called NameValuePair needs to be constructed.
  NameValuePair[] audienceId = new NameValuePair[1];

  NameValuePair nvp = new NameValuePair();
  nvp.setName("CustomerID");
  nvp.setValueDataType("numeric");
        nvp.setValueAsNumeric(2021);

  audienceId[0] = nvp;

        // personalized fields: each personalized field is a name value pair, so again we use the
        // custom type "NameValuePair".  For this example, we want to send two personalized fields
(emailAddress, gender)
 NameValuePair[] personalizedFields = new NameValuePair[2];
```

```
NameValuePair nvp1 = new NameValuePair();
      nvp1.setName("emailAddress");
      nvp1.setValueDataType("string");
      nvp1.setValueAsString("johndoe@foobar.com");

personalizedFields[0] = nvp1;

NameValuePair nvp2 = new NameValuePair();
      nvp2.setName("gender");
      nvp2.setValueDataType("string");
      nvp2.setValueAsString("male");

personalizedFields[1] = nvp2;

// Cell code
String[] cellCodes = { "CC243935" };

// Load the attachment data from the file system using a data source
FileDataSource logo = new FileDataSource(new File("C:\\logo.png"));
DataHandler handler = new DataHandler(logo);
Base64Binary attachmentBinary = new Base64Binary();
attachmentBinary.setBase64Binary(handler);
ContentType_type0 actualContentType = new ContentType_type0();
actualContentType.setContentType_type0(handler.getContentType());
// specify the content type for the attachment
attachmentBinary.setContentType(actualContentType);

// Add the attachment
Attachment attachment = new Attachment();
attachment.setFileName("First Attachment");
attachment.setLabel("Attachment");
attachment.setFileContent(attachmentBinary);

// Configure attachments
Attachment[] attachments = new Attachment[] {attachment};

// Additional Options - this is a name value pair again - but for now
// send as null
NameValuePair[] additionalOptions = null;

NameValuePair[] trackingFields = null;

// locale - rely on default by setting as null;
String locale = null;

/**
       * Calling the Method:
       * 1) set up a connection object with the URL of the TMS webservice
       * 2) Construct the required security header with the authentication credentials
       * 3) Construct the method and Set the parameters
       * 4) Make the call
       * 5) Process the response
 */

// connection object
TMSStub stub = new TMSStub("http://<Replace IP of Deliver TMS Service>:<PORT>/delivertms/services/TMS");
```

```
 ServiceClient serviceClient = stub._getServiceClient();
 serviceClient.getOptions().setProperty(HTTPConstants.SO_TIMEOUT, new Integer(60 * 1000));
 serviceClient.getOptions().setProperty(HTTPConstants.CONNECTION_TIMEOUT, new Integer(60 * 1000));

 // authentication: the TMS web service requires the client to submit
 // user and pw info via soap headers.
 // the following code sets up the authentication credentials that are
 // passed in via the headers.
 UserName un = new UserName();
 un.setUserName(userName);

 Password pwd = new Password();
 pwd.setPassword(password);

 // make the call
 Response response = stub.sendMailing(mailingCode, audienceId, personalizedFields, cellCodes,
additionalOptions,
   attachments, trackingFields, locale, un, pwd);

 // process the response - a customType Response is returned
 // all responses come back with a top level code that indicates whether
 // or not the request was
 // successful (0) or a warning (1) or error (2) occurred. If the request
 // was not successful, the client code
 // should log/alert the issue, and possibly retry the request depending
 // on the issue
 if (response.getStatusCode() == 0) {
  System.out.println("Request to TMS successful");
 } else // an error or warning occurred
```

Chapter 6. About Dynamic Transactional Images

Dynamic transactional images are images specific to a particular individual that you can include with transactional email messages. Dynamic transactional images are sent as attachments to transactional email but they display as images embedded in the body of the message.

A common example of using individualized images involves embedding ticket barcodes or QR codes in a transactional email message. Your corporate business systems create the barcode and provide it to the local transactional email client. The client includes the image attachment in the SOAP request submitted to the Deliver TMS. Because the image is transmitted as an attachment, the email recipient sees the barcode displayed in the email even if the email client is configured to turn images off.

Dynamic transactional images are not available in standard email messages nor are they supported for use within advanced scripts for email.

Using dynamic transactional images in transactional email messages

Using dynamic transactional images requires separate actions by email designers and transactional application developers. Email designers define an image label in the email document to indicate where a dynamic transactional image appears in the email. Application developers configure the local transactional email client to reference the image label in transactional email requests. Corporate transactional systems provide the image content when the transactional mailing runs.

Email designers define dynamic transactional image labels by modifying tags in the HTML template that defines the email. Designers can define image labels with the Deliver Document Composer or by modifying the HTML code directly. Image labels use a syntax reserved for dynamic transactional images. The location of the image label determines how and where a dynamic transactional image appears in the body of an email.

Application developers must ensure that all of the labels that appear in the email are referenced in transactional email requests. The image label and the image content must be part of the SOAP request that the local transactional email client submits to the Deliver TMS. The image labels are case sensitive and must appear in the SOAP request exactly as they are defined in the email.

Defining an image label for dynamic transactional images

You define image labels for dynamic transactional images using a specific format.

Use the following syntax to define the label for a dynamic transactional image in an email document.

```
#include:image_label#
```

The local transactional email client references the value that you define for *image_label* in the SOAP request it submits to the Deliver TMS.

The value for *image_label* is case-sensitive. The image label defined in the email must exactly match the image label provided in the SOAP request submitted to the Deliver TMS. The email marketing team and application developers must develop consistent naming conventions and procedures to ensure that the names match.

In the Document Composer, you can add the label for a dynamic transactional image to an email document by using the image widget. You can also add the label for a dynamic transactional image directly to the HTML code in the template used to create the email.

Note: When you preview an email that contains dynamic transactional images, the image links appear to be broken. This behavior is expected. The images are populated when Deliver receives the image content as part of the transactional email request.

Adding dynamic transactional images using the image widget

You can use the image widget in the Deliver Document Composer to define a dynamic transactional image in an email document. Replace the image URL with the image label for the dynamic transactional image.

Before you begin

The following procedure defines a label to identify the image. The image label is case sensitive. Consult with application developers to ensure that transactional email requests reference the image using the exact same image label.

1. Insert or drag an image widget to a droppable zone in the email document.
2. In the **Image** field, enter the label for the dynamic transactional image in the following format.

   ```
   #include:image_label#
   ```

   Replace *image_label* with a unique name to identify the image. For example,

   ```
   #include:barcode1#
   ```
3. Save the email document.

Adding dynamic transactional images directly to email templates

You can add a dynamic transactional image directly into an email document with an IMG tag in the HTML code of the template.

- Define the IMG tag as follows.

  ```
  <img src="#include:image_label#" alt="alt_text_for_image" border="0" />
  ```
- Replace *image_label* with a unique name to identify the image. As a best practice, provide alternate text for the image.

  Example

  For example:

  ```
  <img src="#include:barcode1#"
  alt="Label for dynamic transactional image" border="0" />
  ```

Adding dynamic transactional images in an image link

You can use a dynamic transactional image to create an image link in a transactional email.

- Define an HREF tag in the email template, as follows.

```
<a href="link_target"><img src="#include:image_label#"
alt="alt_text_for_link" border="0" /></a>
```

- Replace *image_label* with a unique name to identify the image. The target URL is the value for *link_target*. As a best practice, provide alternate text for the image.

  Example

  For example:

```
<a href="www.example.com"><img src="#include:picture1#"
alt="Link to web site" border="0" /></a>
```

How to specify dynamic transactional images in the transactional email request

In the web service request for transactional email, specify dynamic transactional images as email attachments. Attachment size requirements for other types of attachments also apply to dynamic transactional images. Each dynamic transactional image cannot exceed 1 Mb and the total of all attachments cannot exceed 2 Mb.

An image label can appear multiple times in an email but it must appear only once in the SOAP request. If you use the same label in multiple attachments.

Use the `attachments` parameter to specify dynamic transactional images as attachments to transactional email. In the `attachments` parameter, the value for the `label` attribute is the image label that is defined in the email document.

Consider the following example of how to configure the `attachments` parameter to specify dynamic transactional images. Assume that you want to send a transactional email that includes an entrance pass to an upcoming customer conference and a map with directions. You must configure two attachments in the web service request. The first attachment is a QR code that admits the recipient to the conference. The second attachment is a map that provides driving directions from the physical address currently on file for the customer. The following example illustrates how the attachments portion of the web service request might look.

```
// Configure attachments In this example, there are two attachments:
// QRblock and MAP_site
Attachment[] attachments = new Attachment[2];

//This is the first of the two attachments
// Load the attachment data from the file system
     using a data source
FileDataSource QRdataSource = new FileDataSource(new File("C:\\QR.png"));
DataHandler QRhandler = new DataHandler(QRdataSource);
Base64Binary QRattachmentBinary = new Base64Binary();
attachmentBinary.setBase64Binary(QRhandler);
ContentType_type0 QRContentType = new ContentType_type0();
QRContentType.setContentType_type0(QRhandler.getContentType());

// specify the content type for the attachment
QRattachmentBinary.setContentType(QRContentType);

// Add the attachment
Attachment QRblock = new Attachment();
QRblock.setFileName("QR.png");
QRblock.setLabel("PremiumTix_QR");
QRblock.setFileContent(QRattachmentBinary);
```

```
//This is the second of the two attachments
// Load the attachment data from the file system
      using a data source
FileDataSource MAPdataSource = new FileDataSource(new File("C:\\SiteMap.png"));
DataHandler MAPhandler = new DataHandler(MAPdataSource);
Base64Binary MAPattachmentBinary = new Base64Binary();
MAPattachmentBinary.setBase64Binary(MAPhandler);
ContentType_type0 MAPContentType = new ContentType_type0();
MAPContentType.setContentType_type0(MAPhandler.getContentType());

// specify the content type for the attachment
MAPattachmentBinary.setContentType(MAPContentType);

// Add the attachment
Attachment MAP_site = new Attachment();
MAP_site.setFileName("SiteMap.png");
MAP_site.setLabel("Map_directions");
MAP_site.setFileContent(MAPattachmentBinary);


// Set the attachment array
attachments[0] = QRblock;
attachments[1] = MAP_site;
```