TOA TECHNOLOGIES

Outbound API SDK for ETAdirect

Version 4.5

TOA Technologies 2014

Table of Content

1 Introduction

1.1 Document Purpose

The document is to provide understanding of the basic Outbound API goals, its methods, and relevant SOAP transactions.

1.2 Scope of Document

This document primarily describes the API that is used by the ETAdirect Outbound Interface for exchanging information (sending requests and accepting responses) with external systems. It also gives an overview of how the ETAdirect Outbound Interface works.

1.3 Target Audience

The document is intended for developers and programmers working with the ETAdirect Outbound Interface in order to integrate ETAdirect with external systems.

1.4 Glossary

Term	***Explanation***
Activity	entity of the ETAdirect system that represents one segment of work. For example: customer activity, lunch break, assisting another resource, warehouse visit
Message	single piece of data sent by the ETAdirect system
Middleware	software that is used to integrate ETAdirect with external systems. Middleware uses ETAdirect API to interact with ETAdirect.
Resource	Element in the resource tree representing a defined company asset
Resource Tree	Hierarchy of company resources, showing “parent-child” relationships
Route	list of work orders/activities assigned to resource for a day
SOAP	a lightweight protocol for information exchange in a decentralized, distributed environment
SOAP 1.1	Version of SOAP protocol used by ETAdirect. See http://www.w3.org/TR/2000/NOTE-SOAP-20000508/
SOAP Client	application or part of application that sends SOAP requests to SOAP Service
SOAP Service	application or part of application that receives SOAP requests sent by SOAP Client
User	1) person who uses ETAdirect 2) account used by a technician/dispatcher/etc. to log into ETAdirect or by middleware to access ETAdirect
Visit	group of activities related to the same customer that generate one customer notification for a case, instead of one notification for an activity

2 Outbound Interface Overview

Outbound API is used for interaction between the ETAdirect message engine and external Middleware. Middleware is a software that needs to be developed in order to integrate ETAdirect with external system(s).
For integration with Voice Platforms it is recommended to use Voice API rather than Outbound API. Voice API is used to initiate outgoing calls to the customers as well as notify them about the upcoming appointments, and is a better fit for this purpose since a lot of the necessary functionality is already implemented on the ETAdirect side.

2.1 Message Engine

The Message engine is a part of the ETAdirect platform designed for creation of messages and their preliminary processing prior to delivery.
ETAdirect Message Engine is highly configurable and can initiate sending messages triggered by different events that take place in the ETAdirect system (such as activity creation, cancellation, completion, or reassignment; inventory installation, de-installation, exchange or hit, messages initiated by the user of the ETAdirect system) or the state of the system at a specific moment of time (activity is not started on time etc.)

2.2 Creation of Messages by Message Engine

Message Engine has a set of predefined triggers associated with various events in ETAdirect. Once a certain event happens an appropriate trigger is activated and the Message Engine initiates the corresponding Message Scenario.

Message Scenario consists of one or more Starting Steps and zero or more Inner Steps.

- Starting Steps are executed once when the scenario is executed.
- Inner steps (result handlers) are optionally executed to handle results of Starting Steps or other Inner Steps.

A message is generated whenever a step is executed.

Each message step has the "Notification Method" property, which defines where messages are sent. Below is the list of some of the Notification Methods:

- **E-mail** – email notification message – handled internally in ETAdirect
- **Voice** – voice notification message – the message is sent to the internal voice agent (Middleware) which then initiates an outbound voice call to an external voice platform
- **External System** – message sent to Middleware

  NOTE! Only **External System** method is related to the Outbound SDK. Other methods are listed here as part of the Message Engine description.

2.3 Message Status

Any message generated by ETAdirect at any moment of time has a Message Status. Message statuses define the flow of message processing. Message statuses are divided into:
Final:

This status means that the message processing is finished and requires no further processing. ETAdirect will neither send any further requests nor expect any incoming requests regarding this message.

Non-final:

These statuses notify the system that the message processing is not finished.

New:

This status is assigned to a message just created in ETAdirect.

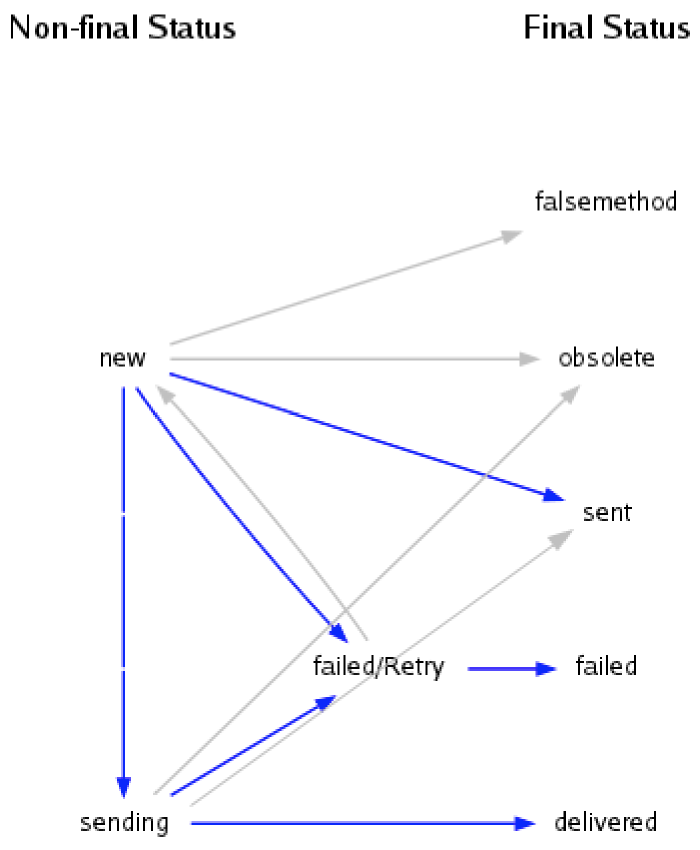
Message Statuses may be changed by Middleware and by ETAdirect internal processes. The processing ends when the message reaches one of the final message statuses.

List of message statuses:

- **falsemethod** (final) – set by ETAdirect if the message itself or an associated object (activity for example) has no fields required for processing by the appropriate method. For example, there is no e-mail address defined for the Email method. This status is not applied to the Outbound API messages.
- **obsolete** (final) – set by ETAdirect if the message is no longer relevant. For example, the day before a message was generated to inform the customer about an activity, but the activity had been cancelled before such message was delivered.
- **delivered** (final) – set by Middleware to signify that the message processing is finished. While "sent" and "delivered" statuses are very similar, in most cases "sent" is used when there is no evidence that the final recipient (person or system) received the message while "delivered" is used when the receipt is confirmed. One of the examples is the voice platform leaving voice mail, in which case the message is typically assigned the "sent" status, but when it identifies that there is human interaction (buttons are pressed), the message status is "delivered".
- **failed** (final unless "attempts" > 1) – set by Middleware to signify that the message delivery has failed, may be set by ETAdirect when an error occurs and the message did not reach its recipient.
- **sent** (final) – set by Middleware to signify that the message was received by Middleware and there will be no further status updates.
- **new** (non-final) – initial status after a message is created, processing has not started.
- **sending** (non-final) – set by Middleware to signify that the message was received by Middleware but its processing is not finished and there will be further status updates.

NOTE: messages with the **"failed"** status can be returned to the **"new"** status and resent for processing (configurable in the 'Step' properties). The 'failed' status is only considered final when there are no retries (default – no retries).

Below is a diagram that reflects state transitions including the final statuses which are set by ETAdirect itself – gray arrows.



Not all statuses may be used in a specific ETAdirect implementation.

3 Workflows

Outbound API supports two kinds of workflow: Simple and Advanced.

Simple workflow is used when Middleware operates in a synchronous mode with ETAdirect, while **Advanced workflow** is used when Middleware operates in an asynchronous mode with ETAdirect.

3.1 Choosing between Simple and Advanced Workflow

This section will help you understand which workflow will fit your project best.

Below are some descriptive features of the Outbound API which we recommend you consider when making your decision.

Throughput: Outbound performance depends on the amount of messages that can be transmitted per second. If this number is too low, then Outbound will not work properly which may result in the systems (ETAdirect and External) getting out of sync.

When Middleware receives a request with several messages, it will not receive other messages until it returns a SOAP response for the first batch.

So if Middleware takes 1 second to process each message, then ETAdirect will not send more than 3600 messages per hour.

For example after a routing run, the routing process generates a message for each activity it moved. If there were 10K activities in a bucket, it would send 10K messages, which would be processed in 3 hours at the rate of 1 message per second.

Batch sending: To achieve faster processing speed, ETAdirect sends messages in batches of 10 or more messages.

This means that if 50 messages are received by Middleware, then quickly put into queue and the response is returned in 1 second, then a single message gets sent effectively in 0.02 seconds.

Middleware internal queue: To achieve faster processing speed, Middleware can implement an internal queue.
When messages arrive at Middleware from ETAdirect, Middleware can quickly put messages into internal queue, then return the SOAP response to ETAdirect immediately. Middleware can then process messages in the queue asynchronously.

This way, 10K messages will be sent from ETAdirect in less than 5 minutes (assuming a batch of 50 messages is put to the queue in 1 second).

Timeouts. If the SOAP response from Middleware is not returned within 30 seconds, then ETAdirect considers it a connection failure, aborts the transaction, and resends the messages later.

For example, if a request contains a batch of 50 messages and each of them takes 1 second to be processed, then the whole request will never be processed – ETAdirect will drop the connection and resend the same messages repeatedly, until they expire in ETAdirect.

So Middleware must guarantee that it returns every SOAP response within 30 seconds.

Message cancellation
For messages that take very long time to be sent away (for example Voice calls), it makes sense to use the Advanced Workflow, even if the result is not relevant to ETAdirect. The reason behind it is that ETAdirect may know that a message is no longer relevant (e.g. the Activity was rescheduled and the Voice call should be postponed/changed).

When using Advanced workflow, ETAdirect sends a "drop_message" request in this case, notifying Middleware that the message no longer needs to be delivered.

This feature can reduce your expenses by canceling obsolete but costly messages and is only available in Advanced workflow.

Scenario	Advanced workflow	Simple workflow	Comments
• Message processing takes long time • ETAdirect is notified of the result at the end of processing	✔	✖	ETAdirect can receive notifications of message processing result in Advanced workflow only.
• Message processing takes long time • ETAdirect is not notified of the result at the end of processing	✔	✔	When using Simple workflow, Middleware must implement internal queue, put messages there, and return response ASAP.
• Message processing is very fast • response to ETAdirect is only returned after all messages are processed	✔	✔	Simple workflow can be used only if processing a batch of messages takes the same amount of time as putting them to the internal queue.
• Message processing takes long time • processing may be cancelled by ETAdirect due to new data	✔	✖	In Advanced workflow, ETAdirect can send a "drop_message" command to notify Middleware that the message is obsolete.

3.2 Simple Workflow (No Delivery Confirmation)

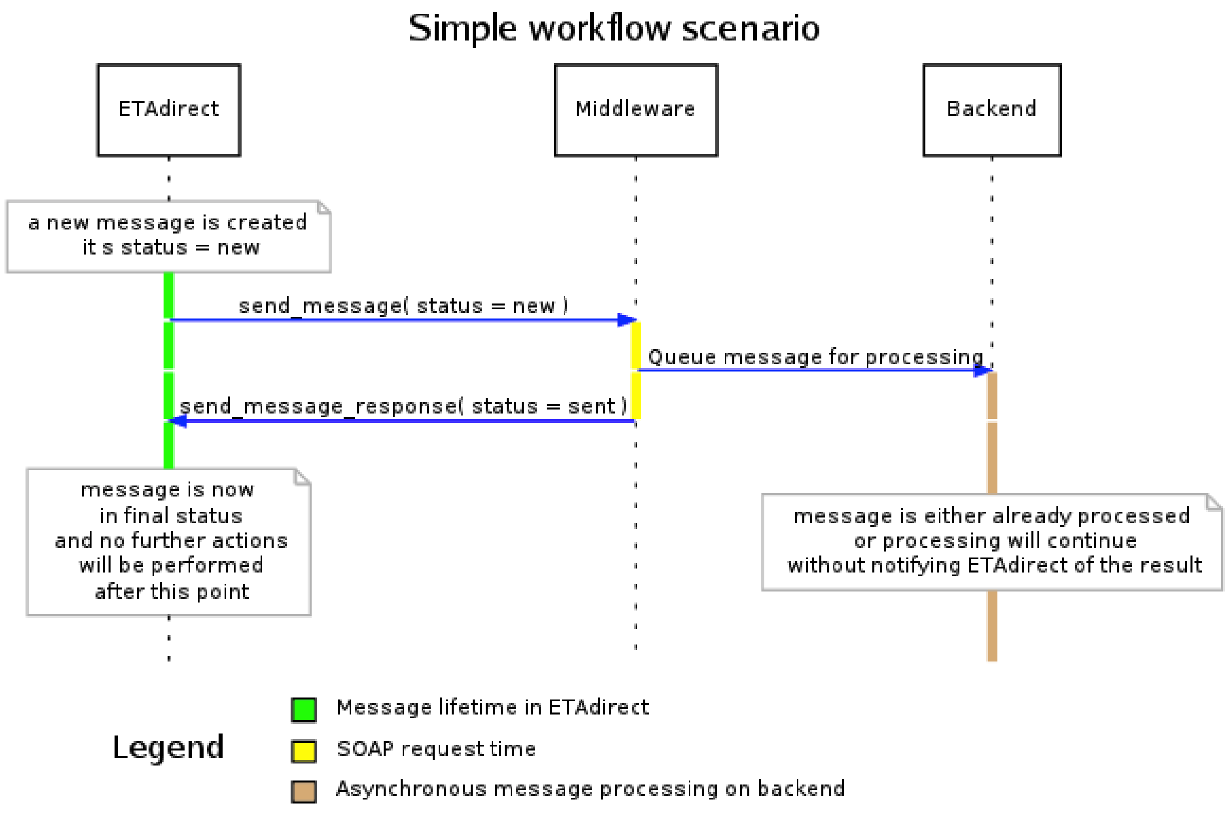
In this workflow Middleware performs the task of delivering messages to the Backend, but does not notify ETAdirect of the result.

Simple workflow:

- ETAdirect sends a "send_message" SOAP request to Middleware.
- The Middleware response contains a final message status: "sent" or "failed".
- ETAdirect will not send any further requests and will not expect any requests regarding this message.
- Middleware guarantees the message delivery to the final recipient.

When the Simple workflow is implemented, special focus should be on the performance of the 'send_message' processing. ETAdirect will not send the next outbound message until it gets the response for the previous message. If Middleware interacts with systems where it has no control over the performance and time delays may appear, this may create a queue of messages on the ETAdirect side.

3.2.1 Sequence Diagram of Simple Workflow


Simple workflow scenario
ETAdirect
Middleware
Backend
a new message is created it s status = new
send_message(status = new)
Queue message for processing
send_message_response(status = sent)
message is now in final status and no further actions will be performed after this point
message is either already processed or processing will continue without notifying ETAdirect of the result
Legend
Message lifetime in ETAdirect
SOAP request time
Asynchronous message processing on backend

3.2.2 Middleware Requirements for Simple Workflow

Middleware must implement a SOAP Service using "**Middleware_Simple.WSDL**" file provided with this SDK.

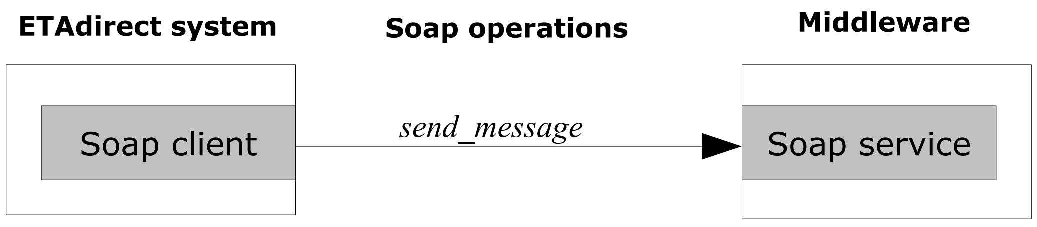
The Middleware SOAP Service will implement one operation:

- **send_message** – this method is called by ETAdirect to send messages to Middleware.

Middleware must respond to **send_message** requests from ETAdirect with one of the following 2 statuses:

- **sent** – message queued for processing
- **failed** – message failed to be queued for processing

3.2.3 Entity Diagram for Simple Workflow



3.3 Advanced Workflow (With Delivery Confirmation)

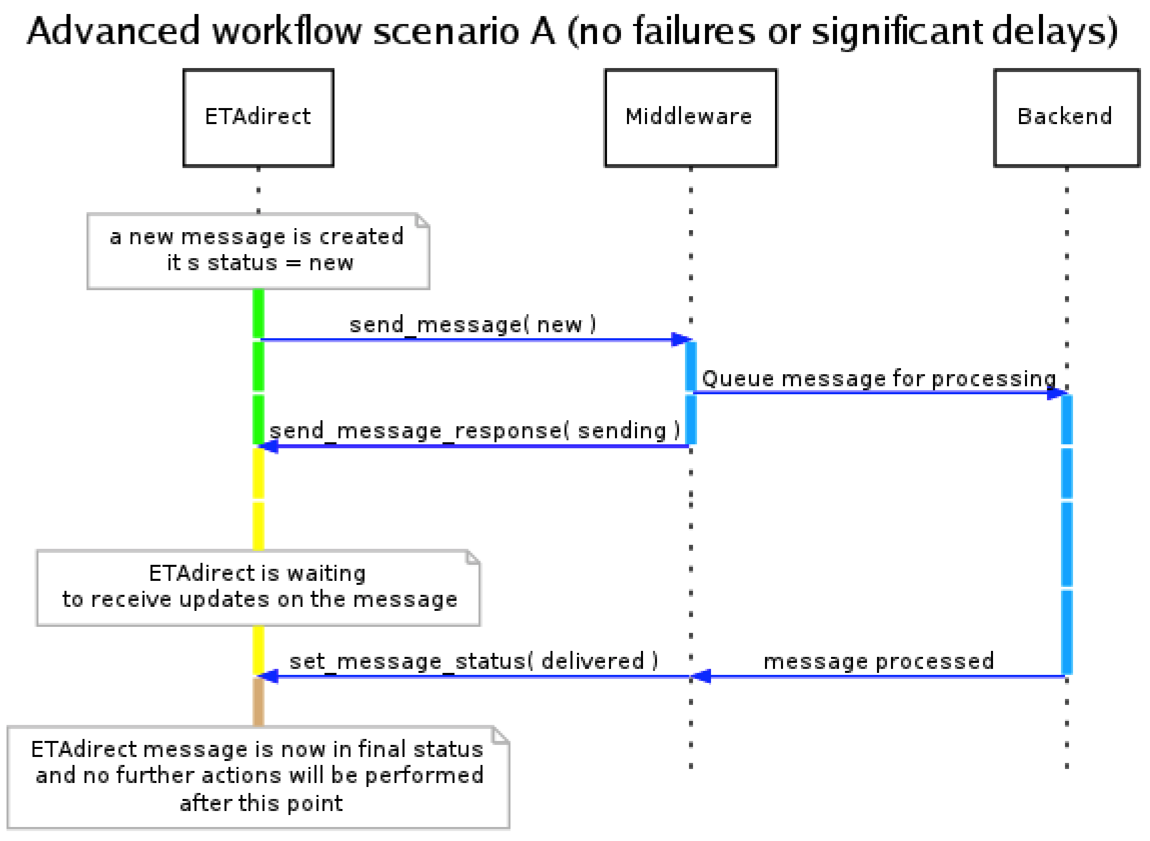
In this workflow Middleware will attempt to deliver messages to the Backend and will notify ETAdirect of the message processing result afterwards.
The Advanced workflow keeps ETAdirect in control of the message processing even after the message was received by Middleware (and up to the moment it is actually processed). This workflow is optimal for the integrations where message delivery takes significant time, for example, integration with IVRs. ETAdirect can generate hundreds of messages but their processing is limited by the number of voice channels available and the call duration.

Advanced workflow:

- ETAdirect sends **send_message** request to Middleware
- The Middleware response contains non-final message status **sending** or final status **failed**
- If **failed** was returned then the processing is finished, otherwise:
- ETAdirect will wait for some time expecting to receive "set_message_status" operation from Middleware
- Middleware should send **set_message_status** request notifying ETAdirect of the result of processing. It should set the final status: **delivered**, **failed,** or **sent**.

3.3.1 Sequence Diagrams of Advanced Workflow

3.3.1.1 Optimistic Scenario: No Failures or Significant Delays


Advanced workflow scenario A (no failures or significant delays)
ETAdirect
Middleware
Backend
a new message is created
it s status = new
send_message(new)
Queue message for processing
send_message_response(sending)
ETAdirect is waiting
to receive updates on the message
set_message_status(delivered)
message processed
ETAdirect message is now in final status
and no further actions will be performed
after this point

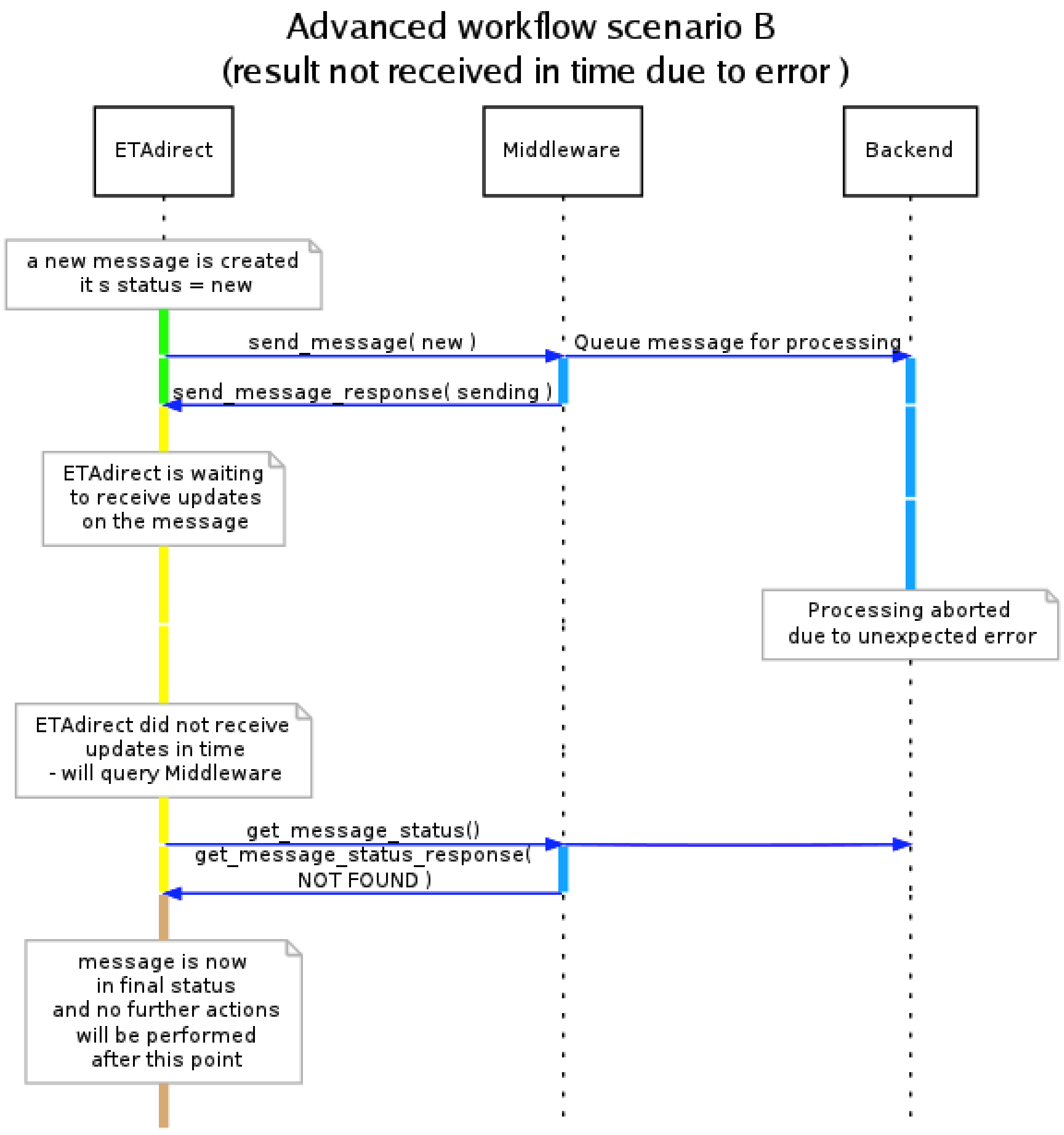
Legend

🟩 message status = "new"

🟨 message status = "sending"

🟫 message status = "delivered"

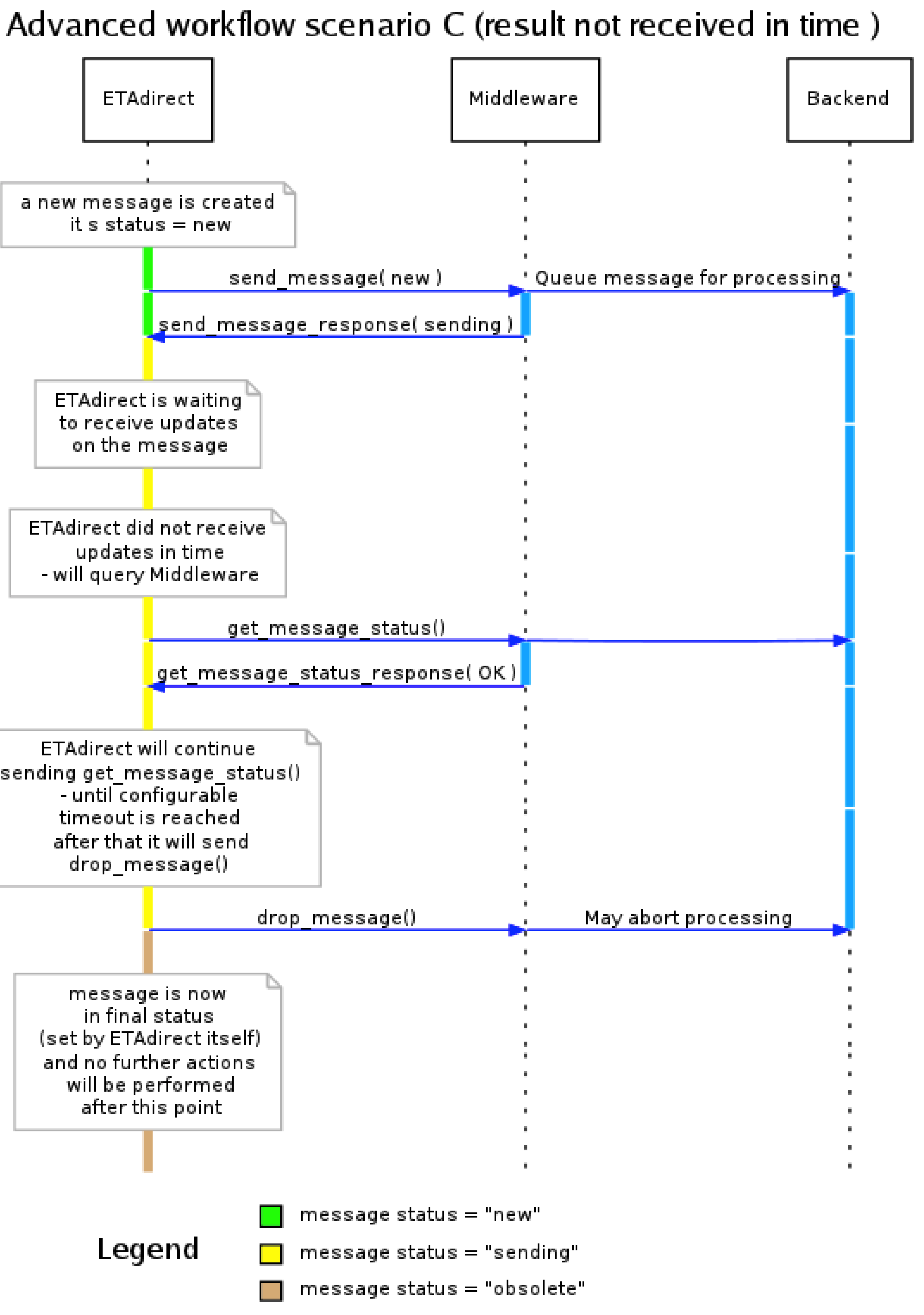
3.3.1.2 Error Scenario: Message Lost



Legend

🟩	message status = "new"
🟨	message status = "sending"
🟫	message status = "failed"

3.3.1.3 Error Scenario: Result not Received in Time


Advanced workflow scenario C (result not received in time)
ETAdirect
Middleware
Backend
a new message is created it s status = new
send_message(new)
Queue message for processing
send_message_response(sending)
ETAdirect is waiting to receive updates on the message
ETAdirect did not receive updates in time - will query Middleware
get_message_status()
get_message_status_response(OK)
ETAdirect will continue sending get_message_status() - until configurable timeout is reached after that it will send drop_message()
drop_message()
May abort processing
message is now in final status (set by ETAdirect itself) and no further actions will be performed after this point
Legend
🟩 message status = "new"
🟨 message status = "sending"
🟫 message status = "obsolete"

3.3.2 Middleware Requirements for Advanced Workflow

3.3.2.1 Middleware SOAP Service for Advanced Workflow

Middleware must implement a SOAP Service using "**Middleware_Advanced.WSDL**" file provided with this SDK.

The Middleware SOAP Service will implement three operations:

- **send_message** – this method is called by ETAdirect to send messages to Middleware.
- **get_message_status** – this method is called by ETAdirect to check if the message is still being processed.
- **drop_message** – this method is called by ETAdirect to indicate that message is obsolete and its processing can be stopped.

The Middleware must respond to **send_message** requests from ETAdirect with one of the following 2 statuses:

- **sending** – message queued for processing
- **failed** – message failed to be queued for processing

The Middleware must respond to **get_message_status** requests from ETAdirect with one of the following codes:

- **OK** if the message is still being processed. ETAdirect will continue sending get_message_status requests periodically
- **NOT FOUND** if the message is not found. ETAdirect will mark this message as "failed"
- **ERROR** if an unexpected error has occurred. ETAdirect will mark this message as "failed"

3.3.2.2 Middleware SOAP Client for Advanced Workflow

Middleware must implement a SOAP Client that connects to ETAdirect at address:

- `https://{INSTANCE}.etadirect.com/soap/outbound/?wsdl`

In this URL {INSTANCE} is a subdomain that may change. For example the integration may be done on one instance while the production will run on another instance. The WSDL contents will be the same at both instances, but the endpoint is different.

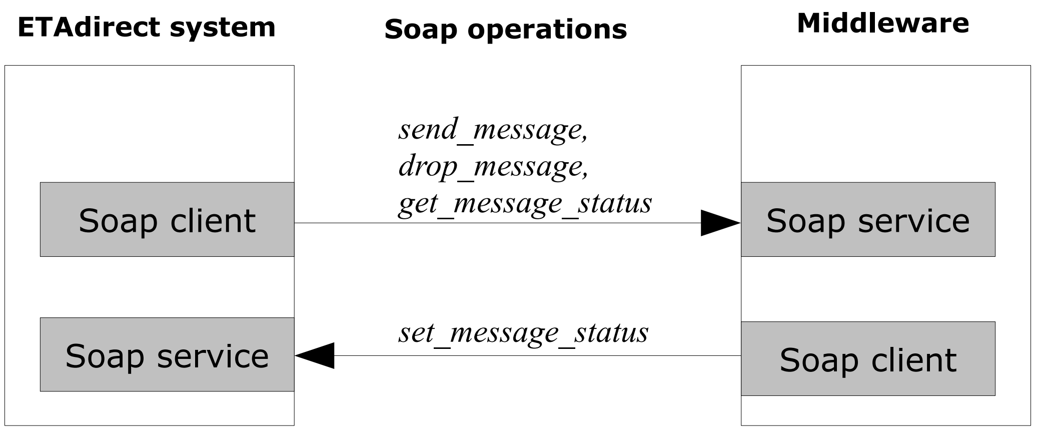
Middleware SOAP Client must send the following operation to ETAdirect:

- **set_message_status** – notify ETAdirect of the message processing result.

The set_message_status should set message status to one of the following final statuses:

- **delivered** – message processed successfully
- **failed** – message processing failed

3.3.3 Entity Diagram for Advanced Workflow



4 Middleware Implementation Guidelines

These guidelines are to help developers pass the API Certification.

4.1 API Certification

Please note that after the initial Middleware development and integration phase, your software will undergo certification to ensure that the integration and API are utilized in an optimal way. The Certification will be based on the Plan of certification that you will receive during the development phase of the integration.

We recommend analyzing the plan of certification before the development is finished to shorten the period of certification.

Below are the client implementation guidelines which should be followed in order to pass certification.

4.2 Middleware Must Return All Responses Immediately

Responses to 'send_message', 'get_message_status', and 'drop_message' must be returned immediately.

When Middleware receives a message via 'send_message' operation and it needs to do some time-consuming processing, Middleware must return the response with "sent", "sending", or "failed" status and then continue with processing in another thread or process.
Middleware implementation must not engage in any long processing before SOAP response has been returned to ETAdirect.

This is because Middleware will not receive any more messages on this message scenario step until it returns the SOAP response. Blocking during the SOAP call means that the messages will be processed very slowly and likely slower than they are generated.

4.3 Middleware Must Support Bulk SOAP Operations

In order to reduce the number of SOAP requests between the ETAdirect platform and Middleware, all methods in the Outbound API support bulk data. That is, each SOAP request can contain the data related to several messages. Also, a response record provides a separate execution result on a per message basis.

It is important when implementing SOAP Service to interpret **`<messages>`** element as array. This may not be noticeable in the initial test with a single message, but the SOAP Service that does not have this feature will fail eventually.

The same applies for Middleware SOAP Client for advanced workflow. It should send 'set_message_status' requests periodically with all messages for a given period (e.g. every few seconds). It should not send each message status individually as it arrives to Middleware.

Example "send_message" bulk request (details omitted for clarity)

```
<env:Envelope>
    <env:Body>
        <send_message xmlns="urn:toatech:agent">
            <user>...</user>
            <messages>
                <message> <!-- message payload --> </message>
                <message> <!-- message payload --> </message>
                <message> <!-- message payload --> </message>
                ... <!-- more messages -->
            </messages>
        </send_message>
    </env:Body>
</env:Envelope>
```

5 Outbound Interface Entities and Structures

5.1 'user' Node

All SOAP requests have a 'user' node used for authentication and described in more details below. When Middleware acts as a server ('send_message', 'drop_message', 'get_message_status') the 'user' can be ignored.

If a 'user' node is present in the request, it should have the following fields:

Name	Type	Description
now	string	current time in ISO 8601 format
company	string	case-insensitive identifier of the Client, for whom the data is to be retrieved (the instance name) Provided by TOA technologies during integration
login	string	case-sensitive identifier of a specific user within the Company. Provided by TOA technologies during integration
auth_string	string	authentication hash auth_string = md5(now + md5(password)), where 'password' is a case-sensitive set of characters used for user authentication provided by TOA technologies during integration

5.1.1 'user' Node and Authentication

Peculiarities of the '*user*' node parameters processing are described for each specific method, though as a rule the node is used for the authentication request. If any of the situations below occur, authentication fails and the relevant error is returned.

Authentication fails if:

1	now	is different from the current time on the server and this difference exceeds the predefined time-window (30 minutes by default)
2	company	cannot be found in ETAdirect
3	login	cannot be found for this company
4	user with this 'login'	is not authorized to use get_capacity method
5	auth_string	is not equal to md5(now+md5(password))
		For example: 'now' = "2011-07-07T09:25:02+00:00" and password = "Pa$$w0rD" then md5 (password) = "06395148c998f3388e87f222bfd5c84b" concatenated string = = "2011-0707T09:25:02+00:0006395148c998f3388e87f222bfd5c84b" **auth_string should be** = "62469089f554d7a38bacd9be3f29a989"

Otherwise authentication is successful and the request is processed further.

6 Detailed Method Description

6.1 'send message' Method

When an internal event or state in the ETAdirect system triggers a new message transaction (for example, an activity is cancelled or not started in time), the ETAdirect system establishes an HTTP connection with the middleware and uses *'send_message'* SOAP method.

NOTE: Keep in mind that 'send_message' transaction execution time is critical, so it is important that 'send_message' does not contain very complex logics so that your system does not create significant delays between the transactions.

NOTE: As the actual data transfer can be rather time-consuming, the Middleware should have an internal queue implemented.

6.1.1 'send_message' Request

Request of *'send_message'* specifies: parent application for the message and message to be sent.

Request of *'send_message'* contains the following parameters:

Field	Required	Description
user	Yes	user node
messages	Yes	sequence of one or more 'message' nodes

6.1.1.1 'send_message' Request 'user' Node Details

Field	Required	Description
user/now	Yes	current time in ISO 8601 format
user/company	-	case-insensitive identifier of the Client, for whom the data is to be retrieved (the instance name) Provided by TOA technologies during integration
user/login	No	may be ignored in *'send_message'* request
user/auth_string	No	may be ignored in *'send_message'* request

6.1.1.2 'send_message' Request 'messages' Array

The *'messages'* array is a set of *'message'* nodes. Each *'message'* node contains message fields. The list of fields is configured in the course of implementation:

Field	Required	Description
app_host app_port app_url	Yes	three fields that define SOAP API location of the calling application. the address may be used to submit message results *('set_messages_status')* **Note:** These fields may be ignored by Middleware as the SOAP API location of ETAdirect endpoint is known beforehand.
message_id	Yes	unique ID of the message in the ETAdirect system used in all other methods to refer to this message integer number that cannot be empty 32-bit integer
address	No	message destination

Field	**Required**	**Description**
		for email – corresponds to the recipient's e-mail address specified for the corresponding message step for voice notification method – corresponds to the phone number source specified for the corresponding message step for external system – the field is meaningless and is filled with constant value '*external_system*'
send_to	No	time limit for sending the message date and time field represented in GMT (*YYYY-MM-DD HH24:MI:SS*); value sent in the field is defined in the message step configuration of ETAdirect and defines the latest time by which message has to be sent agents that deal with a back office system usually should ignore this field, unless it is a part of the solution defined with ETAdirect
subject	No	filled and preprocessed templates for *'subject'* and *'body'* text blocks defined in message step configuration that contain all fields required for the information transfer (activity fields, inventory fields, etc.) format and content of the text blocks are defined in the course of implementation, but can be changed via ETAdirect Manage Application.
body	No	

6.1.1.3 'send_message' Request Example

```
 <soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:urn="urn:toatech:agent">
<soapenv:Header/>
<soapenv:Body>
   <urn:send_message>
      <urn:user>
         <urn:now>2011-11-23T15:50:23+00:00</urn:now>
         <urn:login>user_name</urn:login>
         <urn:company>company_name</urn:company>
         <urn:auth_string>67c5900a04abc54132a52da8a2320be2</urn:auth_string>
      </urn:user>
      <urn:messages>
         <urn:message>
            <urn:app_host>jazz.etadirect.com</urn:app_host>
            <urn:app_port>5900</urn:app_port>
            <urn:app_url>/soap/</urn:app_url>
            <urn:message_id>2006</urn:message_id>
            <urn:address>12345</urn:address>
            <urn:send_to>2011-11-24 01:59:00</urn:send_to>
            <urn:subject></urn:subject>
            <urn:body>{
         "appt_number" : "XXX1234",
         "name":"Rakesh Ivanov",
           "phone": "1234567"
       }</urn:body>
           </urn:message>
        </urn:messages>
     </urn:send_message>
  </soapenv:Body>
</soapenv:Envelope>
```

6.1.2 'send_message' Response

When the middleware accepts the 'send_message' request it has to return the 'message_response'.

Note: Responses to 'send_message' must be returned as soon as possible.

This structure contains the following parameter fields:

Field	Required	Description
message_id	Yes	'message_id' value from the request
status	Yes	new value of the 'status' field. Possible values are: for messages that do not pass validation the status is 'failed' for messages that are correct but do not require actual transfer to the back office system the status is 'sent' for messages that require data transfer to the back office system the status is 'sending' (and the messages are placed on the internal queue of the middleware) for messages successfully delivered to the back office system the status is 'delivered' **Note:** This is a standard set of statuses to be returned and their sending conditions, but for each project it should be agreed with implementation
description	No	new value of the 'description' field; customer-specific additional value that along with Message Status can influence the flow of a message scenario. For example Message Status 'Failed' can differ subject to its descriptions such as 'no_answer' and 'hang_up'. **Note:** Set of possible descriptions should be agreed with implementation
data	No	new value of the 'data' field – used only if so required by ETAdirect solution (e.g. can be used to assign values to activity and inventory properties, cancel activities and set them as 'non-scheduled' as described in more details below)
external_id	No	new value of 'external_id' field – identifier of the message in the internal queue (usually not used)
duration	No	new value of 'duration' field for the message record in ETAdirect (usually not used)

Field	Required	Description
'sent'	No	actual time when the message was sent: GMT YYYY-MM-DDTHH24:MI:SS+00:00 format date and time
'fault_attempt'	No	number of the remaining attempts to resend the message in case the notification has failed this way the external system can change the number of the remaining attempts (e.g. stop or continue resending until success) unless there is a particular need to use the functionality, the field should be omitted, so that the fault attempt logic remains in accordance with the corresponding message scenario step **Note:** Resending in this case does not create a new message, the same message (with the same id) is being resent.
'stop_further_attempts'	No	if '1', notification attempts are stopped 'stop_further_attempts'=1 is equivalent to 'fault_attempt'=0
'time_delivered_start'	No	time delivered interval (promised to the customer) in HH:MM:SS format if 'status' returned is 'delivered' or 'sent', the fields are updated for activity/visit
'time_delivered_end'	No	

6.1.2.1 'send_message' Response Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:urn="urn:toatech:agent">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:send_message_response>
         <urn:message_response>
            <urn:message_id>2006</urn:message_id>
            <urn:status>sent</urn:status>
            <urn:description>everything is fine</urn:description>
         </urn:message_response>
      </urn:send_message_response>
   </soapenv:Body>
</soapenv:Envelope>
```

6.2 'drop_message' Method

'drop_message' method is used to remove messages from the agent internal queue, if message sending should be canceled (e.g if the activity has been canceled or deleted).

Note: Sometimes a situation may occur when there is no real need for the method. Not to change the workflow, a simple method can be implemented that always returns an error.

6.2.1 'drop_message' Request

Request of drop_message specifies message to be dropped and contains the following fields:

Field	Vis	Description
user	M	user node
messages	M	sequence of one or more 'message' nodes

6.2.1.1 'drop_message' Request 'user' Node Details

The 'user' node contains the following fields:

Field	Description
now	current time in ISO 8601 format
company	case-insensitive identifier of the Client, for whom the data is to be retrieved (the instance name) Provided by TOA technologies during integration
login	may be ignored in drop_message request
auth_string	may be ignored in drop_message request

6.2.1.2 'drop_message' Request 'messages' Array

The *'messages'* array is a set of *'message'* nodes. Each *'message'* node contains just one field:

Field	Required	Description
message_id	Yes	ID of the message to be removed

6.2.1.3 'drop_message' Request Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" [^]
xmlns:urn="urn:toatech:agent">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:drop_message>
         <urn:user>
            <urn:now>2011-11-23T15:50:23+00:00</urn:now>
            <urn:login>user_name</urn:login>
            <urn:company>company_name</urn:company>
            <urn:auth_string>67c5900a04abc54132a52da8a2320be2</urn:auth_string>
         </urn:user>
         <urn:messages>
            <urn:message>
               <urn:message_id>2006</urn:message_id>
            </urn:message>
            <urn:message>
               <urn:message_id>2007</urn:message_id>
            </urn:message>
         </urn:messages>
      </urn:drop_message>
   </soapenv:Body>
</soapenv:Envelope>
```

6.2.2 'drop_message' Response

The 'drop_message' response is an array of one or more 'message_response' nodes.

Note: Responses to 'drop_message' must be returned as soon as possible.

Each 'message response' node contains the following fields:

Field	Required	Description
message_id	Yes	ID of the message
result	Yes	transaction result description
result/code	Yes	message removal result code. Possible values are: • OK – message successfully removed • NOT FOUND – message ID unknown for the agent • ERROR – either the message is under processing at the moment or an internal agent error occurred
result/desc	No	error description

6.2.2.1 'drop_message' Response Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" [^]
xmlns:urn="urn:toatech:agent">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:drop_message_response>
         <urn:message_response>
            <urn:message_id>2006</urn:message_id>
            <urn:result>
               <urn:code>OK</urn:code>
            </urn:result>
         </urn:message_response>
         <urn:message_response>
            <urn:message_id>2007</urn:message_id>
            <urn:result>
               <urn:code>ERROR</urn:code>
               <urn:desc>Cannot drop the message. The message is under processing at 
the moment.</urn:desc>
            </urn:result>
         </urn:message_response>
      </urn:drop_message_response>
   </soapenv:Body>
</soapenv:Envelope>
```

6.3 'get_message_status' Method

'get_message_status' method is used to retrieve a message status from the agent internal queue (when the message handling status has not been returned to ETAdirect in time).

6.3.1 'get_message_status' Request

'get_message_status' request specifies the message for which the status is to be retrieved and contains the following fields:

Field	Required	Description
user	Yes	user node
messages	Yes	array of one or more message nodes

6.3.1.1 'get_message_status' Request 'user' Node Details

The '*user*' node contains the following fields:

Field	Description
now	current time in ISO 8601 format
company	case-insensitive identifier of the Client, for whom the data is to be retrieved (the instance name) Provided by TOA technologies during integration
login	may be ignored in get_message_status request
auth_string	may be ignored in get_message_status request

6.3.1.2 'get_message_status' Request 'messages' Array

The 'messages' array is a set of 'message' nodes. Each 'message' node contains just one field:

Field	Required	Description
message_id	Yes	ID of the message the status of which is to be returned

6.3.1.3 'get_message_status' Request Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" [^]
xmlns:urn="urn:toatech:agent">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:get_message_status>
         <urn:user>
            <urn:now>2011-11-23T15:50:23+00:00</urn:now>
            <urn:login>user_name</urn:login>
            <urn:company>company_name</urn:company>
            <urn:auth_string>67c5900a04abc54132a52da8a2320be2<</urn:auth_string>
         </urn:user>
         <urn:messages>
            <urn:message>
               <urn:message_id>2006</urn:message_id>
            </urn:message>
            <urn:message>
               <urn:message_id>2007</urn:message_id>
            </urn:message>
            <urn:message>
               <urn:message_id>2008</urn:message_id>
            </urn:message>
         </urn:messages>
      </urn:get_message_status>
   </soapenv:Body>
</soapenv:Envelope>
```

6.3.2 'get_message_status' Response

'get_message_status' message response is an array of one or more 'message_response' nodes.

Note: Responses to 'get_message_status' must be returned as soon as possible.

Each 'message_response' node contains the following fields:

Field	Required	Description
message_id	Yes	ID of the message
result	Yes	node that contains transaction result description

6.3.2.1 'get_message_status/result' Node

Each result node contains the following elements

Field	Required	Description
result/code	Yes	message removal (from the internal middleware queue) result code. Possible values are: NOT FOUND – message ID is unknown to the agent OK (desc = WAITING) – message sending has not yet been started OK (desc = SENDING) – message is being processed at the moment ERROR – an internal agent error occurred
result/desc	No	code description

6.3.2.2 'get_message_status' Response Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" [^]
xmlns:urn="urn:toatech:agent">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:get_message_status_response>
         <urn:message_response>
            <urn:message_id>2006</urn:message_id>
            <urn:result>
               <urn:code>OK</urn:code>
               <urn:desc>WAITING</urn:desc>
            </urn:result>
         </urn:message_response>
         <urn:message_response>
            <urn:message_id>2007</urn:message_id>
            <urn:result>
               <urn:code>OK</urn:code>
               <urn:desc>SENDING</urn:desc>
            </urn:result>
         </urn:message_response>
         <urn:message_response>
            <urn:message_id>2006</urn:message_id>
            <urn:result>
               <urn:code>OK</urn:code>
               <urn:desc>NOT FOUND</urn:desc>
            </urn:result>
         </urn:message_response>
      </urn:get_message_status_response>
   </soapenv:Body>
</soapenv:Envelope>
```

6.4 'set_message_status'

'set_message_status' method is the only method used by ETAdirect SOAP API. The method returns transaction results.

If as a result of 'send_message' method, Middleware has returned status = 'sending', ETAdirect SOAP API method 'set_message_status' is used to return the result after the actual end of the transaction. Middleware can also use this method to update fields of the message in the ETAdirect system. One call of this method can be used to set the status for several messages.

6.4.1 'set_message_status' Request

'set_message_status' request defines the message status data to be returned and contains the following fields:

Field	Required	Description
user	Yes	user node
messages	Yes	array of one or more 'message' nodes

6.4.1.1 'set_message_status' Request 'user' Node Details

The 'user' node contains the following fields:

Field	Description
now	current time in ISO 8601 format
company	case-insensitive identifier of the Client, for whom the data is to be retrieved (the instance name) Provided by TOA technologies during integration
login	case-sensitive identifier of a specific user within the Company provided by TOA technologies during integration
auth_string	authentication hash auth_string = md5(now + md5(password)) where 'password' is a case-sensitive set of characters used for user's authentication provided by TOA technologies during integration

6.4.1.2 'set_message_status' Request 'messages' Array

'messages' array is a set of one or more 'message' nodes. Each 'message' node contains:

Field	Required	Description
message_id	Yes	'message_id' value from the 'send_message' request
status	Yes	new value of the 'status' to be set for the message; possible **case-sensitive** values are: **sending** – message still in the internal queue of the Middleware waiting to be delivered (another call of 'set_message_status' will be required later to set the final status of the delivery)

Field	Required	Description
		delivered – message successfully transferred to the back office system **failed** – transaction failed **sent** – message sent but there is no way to confirm that it has reached the final recipient (for example E-mails) **Note:** These are the default statuses to be returned, though for each specific project they may be agreed at the implementation phase.
description	No	new value of the 'desc' field customer-specific additional value that along with Message Status can influence the flow of a message scenario e.g Message Status 'Failed' can differ subject to its descriptions such as 'no_answer' and 'hang_up'
data	No	new value of the 'data' field
external_id	No	ID of the message in the external system
duration	No	new value of the 'duration' field
sent	No	new value of the 'sent' field
fault_attempt	No	number of the remaining attempts to resend the message in case the notification has failed this way the external system can change the number of the remaining attempts (e.g.stop or continue resending until success) unless there is a particular need to use the functionality, the field should be omitted, so the fault attempt logic remains in accordance to the corresponding message scenario step **Note:** Resending in this case does not create a new message, the same message (with the same id) is being resent.
stop_further_attempts	No	if '1', notification attempts are stopped 'stop_further_attempts'=1 is equivalent to 'fault_attempt'=0
time_delivered_start	No	time delivered interval (promised to the customer) in HH:MM:SS format if 'status' returned is 'delivered' or 'sent', the fields are updated for activity/visit
time_delivered_end	No	

6.4.1.3 'set_message_status' Request Example

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" [^]
xmlns:urn="urn:toa:outbound">
   <soapenv:Header/>
   <soapenv:Body>
      <urn:set_message_status>
         <user>
            <now>2011-11-23T15:50:23+00:00</now>
            <login>user_name</login>
            <company>company_name</company>
            <auth_string>67c5900a04abc54132a52da8a2320be2</auth_string>
         </user>
         <messages>
            <message>
               <message_id>2006</message_id>
               <status>failed</status>
               <description>WRONG_TIME</description>
               <data>Night time</data>
             </message>
            <message>
               <message_id>2007</message_id>
               <status>delivered</status>
               <description>COMPLETED</description>
               <data></data>
               <duration>14</duration>
               <sent>2011-11-29T12:54:22+00:00</sent>
             </message>
             <message>
               <message_id>2008</message_id>
               <status>failed</status>
               <description>WRONG_TIME</description>
               <data>Night time</data>
             </message>
         </messages>
      </urn:set_message_status>
   </soapenv:Body>
</soapenv:Envelope>
```

6.4.2 'set_message_status' Response

'set_message_status' response is an array of *'message_response'* nodes. Each *'message_response'* node contains the following elements:

Field	Required	Description
message_id	Yes	ID of the message
result	Yes	transaction result description
result/code	Yes	Message status retrieval result code. Possible values are:

		• NOT FOUND – message ID is unknown • OK – message has been updated • ERROR – an internal agent error occurred
result/desc	No	error description

6.4.2.1 'set_message_status' Response Example

```
<SOAP-ENV:Body>
  <ns1:set_message_status_response>
    <message_response>
      <message_id>2006</message_id>
      <result>
        <code>OK</code>
      </result>
    </message_response>
    <message_response>
      <message_id>2007</message_id>
      <result>
        <code>OK</code>
      </result>
    </message_response>
    <message_response>
      <message_id>2007</message_id>
      <result>
        <code>NOT FOUND</code>
      </result>
    </message_response>
  </ns1:set_message_status_response>
</SOAP-ENV:Body>
```

6.5 Updating Properties and Processing Activities with 'data'

A message processing result returned by an agent in the send_message response or via the set_message_status call, can be processed by the ETAdirect system to perform the following actions:

- update all custom properties of activity and inventory, as well as a specific set of activity fields
- cancel activities and set non-scheduled activities

The fields to be assigned and the corresponding values are passed in the 'data' field.

The **#params?** string is used as a delimiter between 'data' itself and the passed parameters. The format of the parameter line is similar to URL. The **&** character is used as a delimiter between different parameters. Names and values of parameters are URL encoded.

6.5.1 Updating Fields and Properties

All custom properties of inventory and activity can be updated with the agent's response:

- To update the custom property the 'data' node should contain the following string:

  #params?*property label=value to be set*

- For example to set 'cconfirmed' property to '1' the data should contain:

  #params?cconfirmed=1

Only a predefined set of activity fields can be updated with the agent's response:

- The fields that can be updated are (values in the list a labels of the fields):
  - email
  - sms
  - cell (synonym for 'sms')
  - phone
  - appt_number
  - customer_number
  - customer_name
  - address
  - city
  - state
  - zip
- To update the field from the list, the 'data' node should contain the following string:

  #params?*field label=value to be set*

- For example to set 'phone' field to '123456' the data should contain:

  #params?phone=123456

Note: fields 'address', 'city', 'state' and 'zip' are used by geocoding and, therefore, must contain valid values of the customer's address, city of residence, state and zip/post code. Other values will not be resolved correctly by the geocoding server.

6.5.2 Managing Activities

Activities can be cancelled or set unscheduled with the agent's response:

- To cancel an activity the 'data' node should contain the following string:

#params?action=cancel_activity

- To set an activity unscheduled the 'data' node should contain the following string:

  #params?action=unschedule_activity

6.5.3 Bulk Action

One middleware response can contain several updates, delimited with the '**&**' sign. For example, to set 'cconfirmed' property to '1', 'phone' filed to '123456' and make activity unscheduled, the 'data' node should contain the following string:

#params?cconfirmed=1&phone=123456&action=unschedule_activity

Note: The total length of the 'data' field cannot exceed 255 characters. If a submitted 'data' value exceeds the limit, it can be correctly processed but will be truncated in the database.

7 Previous Versions

The Outbound API in ETAdirect 4.5 is fully compatible with the Outbound API of 4.2, 4.3 and 4.4 versions. The only change is that the 'device' field of the message engine transactions has gone obsolete. If sent, the field will be ignored.

8 Appendix A – Middleware_Simple.WSDL

The file "**Middleware_Simple.WSDL**" should be provided as part of the SDK. This file is referenced in section 3.2.2.

9 Appendix B – Middleware_Advanced.WSDL

The file "**Middleware_Advanced.WSDL**" should be provided as part of the SDK. This file is referenced in section 3.3.2.2.