


[← return to chatwoot/chatwoot](#)

## How to manually trigger WhatsApp template syncing if I am running Chatwoot in Docker

⚡ Fast

Searched across chatwoot/chatwoot

 Go deeper

You're asking about manually triggering WhatsApp template syncing when running Chatwoot in Docker.

### Answer

To manually trigger WhatsApp template syncing in a Docker environment, you need to access the Rails console and call the `sync_templates` method on your WhatsApp channel.

### Steps for Docker Environment

1. Access the Rails container console:

```
docker-compose exec rails bundle exec rails console
```

`docker-compose.yml:44`

2. Find your WhatsApp channel and trigger sync:

```
# Find your WhatsApp channel
channel = Channel::WhatsAppChannel.find_by(provider: 'whatsapp')
# or
channel = Channel::WhatsAppChannel.find_by(provider: 'whatsapp')

# Trigger template sync
channel.sync_templates
```

Ask a follow-up question

Deep Research 

### How Template Syncing Works

The template syncing functionality differs based on your WhatsApp provider:

**For WhatsApp Cloud API:** The `sync_templates` method fetches templates from Facebook's Graph API and handles pagination automatically.

`whatsapp_cloud_service.rb:29-34`

**For 360Dialog:** The sync fetches templates from their API endpoint and updates the channel's message templates. `whatsapp_360_dialog_service.rb:27-32`

Both implementations update the `message_templates` field and set `message_templates_last_updated` timestamp, even if the API request fails to prevent continuous retry attempts. `whatsapp_cloud_service.rb:30-31`

### Alternative: Using Sidekiq Console

If you prefer using the background job system:

```
docker-compose exec sidekiq bundle exec rails console
```

`docker-compose.yml:62`

Then execute the same Ruby commands to trigger the sync.

### Notes

The template syncing process is designed to be safe - it marks templates as updated before attempting the sync to prevent infinite retry loops if the provider configuration is incorrect. `whatsapp_cloud_service.rb:30-31` The sync methods are tested to handle both successful responses and API failures gracefully. `whatsapp_cloud_service_spec.rb:224-231`

Wiki pages you might want to explore:

- [Overview \(chatwoot/chatwoot\)](#)

 Copy Response Share! chatwoot/chatwoot `docker-compose.yml`

```
41 - NODE_ENV=development
42 - RAILS_ENV=development
43 entrypoint: docker/entrypoints/rails.sh
44 command: ["bundle", "exec", "rails", "s", "-p", "3000", "-b", "0.0.0.0"]
45
46 sidekiq:
47 <<: *base
```

```
26 process_response(response)
27 end
28
29 def sync_templates
30 # ensuring that channels with wrong provider config wouldn't keep trying
31 whatsapp_channel.mark_message_templates_updated
32 templates = fetch_whatsapp_templates("#{business_account_path}/message_templates")
33 whatsapp_channel.update(message_templates: templates, message_templates_last_updated: Time.now)
34 end
35
36 def fetch_whatsapp_templates(url)
37 response = HTTParty.get(url)
```

← return to mautic/mautic

# How to manually trigger processing of contact export

⚡ Fast

Searched across mautic/mautic

Go deeper

You're asking about manually triggering the processing of contact exports in Mautic.

Based on the codebase, contact exports in Mautic can be processed through a scheduled export system. [ContactScheduledExportCommand.php:20-25](#) The system uses the `ContactScheduledExportCommand` which can be executed manually to process pending contact exports.

## Manual Triggering Methods

### 1. Command Line Execution

You can manually trigger contact export processing using the console command: `ContactScheduledExportCommand.php:24`

```
php bin/console mautic:contacts:scheduled_export
```

The command supports an optional `--ids` parameter to process specific export jobs: [ContactScheduledExportCommand.php:39-44](#)

`php bin/console mautic` Ask a follow-up question

Deep Research

### 2. Background Process

The system has a configuration setting for background CSV export processing. [messages.ini:398-399](#) When enabled, contact exports are automatically queued for background processing.

### 3. Export Processing Flow

The command retrieves scheduled exports from the database: [ContactScheduledExportCommand.php:55-60](#) If specific IDs are provided, it processes those exports; otherwise, it picks up to 10 scheduled exports from the queue.

The processing involves dispatching events through the event system: [ContactScheduledExportCommand.php:28](#) This allows other parts of the system to hook into the export process.

## Export Status and Notifications

When exports are processed in the background, users receive email notifications with download links: [messages.ini:1092-1095](#) The system shows a "File is queued for processing" message and sends the completed export to the user's email address.

### Notes

The contact export system integrates with Mautic's broader reporting and data export infrastructure. The `ReportModel` class also handles CSV exports for reports, but this is separate from the contact-specific export functionality. The JavaScript code in `1a.content.js` handles the frontend aspects of CSV export processing, including background export detection and AJAX handling for the export process.

Wiki pages you might want to explore:

- [Contact Management System \(mautic/mautic\)](#)

Copy Response Share

```

mautic/m... app/bundles/LeadBundle/Command/ContactScheduledExp...
17 use Symfony\Component\Console\Output\OutputInterface;
18 use Symfony\Component\EventDispatcher\EventDispatcherInterface;
19
20 class ContactScheduledExportCommand extends Command
21 {
22     private const PICK_SCHEDULED_EXPORTS_LIMIT = 10;
23
24     public const COMMAND_NAME = 'mautic:contacts:schedule
25
26     public function __construct(
27         private ContactExportSchedulerModel $contactExportSchedulerModel,
28         private EventDispatcherInterface $eventDispatcher,
29         private FormatterHelper $formatterHelper,
30         private ProcessSignalService $processSignalService,
31     ) {
32
33     }
34
35     {
36         $this
37             ->setName(self::COMMAND_NAME)
38             ->addOption(
39                 '--ids',
40                 null,
41                 InputOption::VALUE_REQUIRED,
42                 'Comma separated contact_export_scheduler ids.'
43             );
44     }
45
46     parent::configure();
47 }
48
49
50     fn (int $signal) => $output->writeln(sprintf('Signal %d caught
51
52
53
54
55     $ids = $this->formatterHelper->csvToJson($input->getOption(

```