Import / Export — Backup and Restore

Blogr's import/export system allows you to back up all blog data (posts, categories, tags, series, CMS pages, users, media) as well as data from installed extensions.

Export Format

Export files are JSON (`.json`) or ZIP (`.zip`) when media files are included.

`data.json` structure:

Happytodev

```
{
  "format_version": "2.0",
  "version": "1.23.17",
  "exported_at": "2026-07-04T12:00:00+00:00",

  "posts":                    [],
  "post_translations":        [],
  "series":                   [],
  "series_translations":      [],
  "categories":               [],
  "category_translations":    [],
  "tags":                     [],
  "tag_translations":         [],
  "user_translations":        [],
  "post_translation_categories": [],
  "post_translation_tags":    [],
  "users":                    [],
  "cms_pages":                [],
  "cms_page_translations":    [],
  "media_files":              [],

  "extension_states": {
    "my-extension":   { "disabled_at": null },
    "other-ext":      { "disabled_at": "2026-07-01T12:00:00+00:00" }
  },

  "extensions": {
    "my-extension": {
      "version": "1.0.0",
      "data": { … },
      "media_files": [ "my-extension/images/foo.jpg" ]
    }
  }
}
```

Happytodev

Format Fields

Field	Description
`format_version`	Export format version (`1.0` = without extensions, `2.0` = with extensions)
`version`	Blogr package version used for the export
`extension_states`	Enable/disable state of each extension at export time

Field	Description
`extensions`	Extension-specific data, keyed by extension identifier

Commands

```
# Export all data (JSON)
php artisan blogr:export

# Export with media files (ZIP)
php artisan blogr:export --include-media

# Export to a custom path
php artisan blogr:export --output=/tmp/backup.json

# Export only a specific extension's data
php artisan blogr:export --only=extensions.my-extension

# Export multiple targeted sections
php artisan blogr:export --only=posts --only=extensions.blogr-docs

# Export everything except some sections
php artisan blogr:export --skip=tags --skip=media_files

# Import a backup file
php artisan blogr:import /tmp/backup.json

# Import with overwrite
php artisan blogr:import /tmp/backup.json --overwrite

# Import only specific sections from a full backup
php artisan blogr:import /tmp/backup.json --only=extensions.blogr-docs

# Import everything except categories and tags
php artisan blogr:import /tmp/backup.json --skip=categories --skip=tags

# Restore after a crash (validation + import)
php artisan blogr:recover /tmp/backup.zip

# Validate backup integrity without importing
php artisan blogr:recover /tmp/backup.json --dry-run
```

Happytodev

Selective Export / Import

The `--only` and `--skip` options let you choose exactly which sections to include or exclude from an export or import.

Supported Keys

Key	Targeted Section
`posts`	Posts
`post_translations`	Post translations
`series`	Series
`categories`	Categories
`tags`	Tags
`users`	Users
`cms_pages`	CMS pages
`extension_states`	Extension enable/disable state
`extensions`	All extensions
`extensions.my-plugin`	A specific extension
`media_files`	Media files

Happytodev

Usage Examples

```
# Develop plugin documentation offline, then import on production
php artisan blogr:export --only=extensions.blogr-docs --output=docs-backup.json
php artisan blogr:import docs-backup.json --only=extensions.blogr-docs

# Migrate only posts between two instances
php artisan blogr:export --only=posts,post_translations --output=posts-only.json
php artisan blogr:import posts-only.json --overwrite

# Sync extensions without touching content
php artisan blogr:export --only=extensions,extension_states
php artisan blogr:import backup.json --only=extensions,extension_states
```

Rules

- `--only` and `--skip` are mutually exclusive. If `--only` is set, `--skip` is ignored.
- Without `--only` or `--skip`, all sections are included (default behavior, backward compatible).
- Nested keys use dot notation: `extensions.blogr-docs` to target a specific extension.

Happytodev

Backward Compatibility

- Files in format `1.0` (without `extension_states` or `extensions`) are imported normally — missing sections are silently ignored.
- Files in format `2.0` imported on an older Blogr version have their unknown sections ignored.
- The `--only` and `--skip` options work with all formats.

Developing an Extension with Backup Support

For an extension to participate in backups, it must implement the `ExportableExtension` interface:

```php
use Happytodev\Blogr\Contracts\ExportableExtension;

class MyExtension implements ExportableExtension
{
    // … required BlogrExtension methods …

    public function getExportKey(): string
    {
        return 'my-extension';
    }

    public function getExportData(): array
    {
        return [
            'settings' => SettingModel::all()->toArray(),
            'stats'    => StatsModel::all()->toArray(),
        ];
    }

    public function importData(array $data, array $options): array
    {
        $imported = 0;
        foreach ($data['settings'] ?? [] as $row) {
            SettingModel::updateOrCreate(…);
            $imported++;
        }
        return ['imported' => $imported, 'skipped' => 0];
    }

    public function getExportMediaPaths(): array
    {
        return SettingModel::pluck('image')->filter()->toArray();
    }
}
```

Happytodev

Crash Recovery

After data loss (server crash, database reset), the recovery workflow is:

```
1. php artisan blogr:recover latest-backup.zip
2. The service validates file integrity
3. It lists expected vs installed extensions
4. It runs migrations if needed
5. It restores data + media + extension states
6. It clears the cache
```

The `blogr:recover` command can be integrated into your deployment playbooks or CI/CD scripts for automated restoration.

Happytodev