Languages (new concept)

Overview

> ⚠️ The new language concept will be released with version 4.3+

TeamCal Neo 4.3+ has modernized its language system from 4 large files to a more efficient controller-specific structure. This guide helps you migrate your custom language files to the new system while maintaining full compatibility.

🚀 Quick Start

Good News: Your existing language files still work! The new system automatically falls back to your current files if the new structure isn't available.

Migration is Optional: You can continue using your current 4-file structure indefinitely. The new system only activates when you set `USE_SPLIT_LANGUAGE_FILES` to true in config.app.php.

What's Changed

Old System (Still Supported)

```
languages/
├── english.php          # Framework strings (~1,026 keys)
├── english.app.php      # Application strings (~730 keys)
├── english.gdpr.php     # GDPR strings (~13 keys)
└── english.log.php      # Logging strings (~103 keys)
```

New System (Optional Performance Upgrade)

```
languages/
└── english/              # Language folder
    ├── core.php          # Essential strings (~26 keys)
    ├── about.php         # About page strings (~8 keys)
    ├── absence.php       # Absence management (~94 keys)
    ├── calendar.php      # Calendar features (~89 keys)
    ├── user.php          # User management (~45 keys)
    └── ...               # 35 total controller files
```

Benefits of Migration

Performance Improvements

- **80-99% Memory Reduction**: Load only 26-355 keys instead of 1,872 keys per page
- **Faster Loading**: 2-4 small files instead of 4 large files per request
- **Better Organization**: Find language keys by feature/controller

Developer Experience

- **Familiar Structure**: Same PHP `$LANG` array syntax you already know
- **Logical Organization**: Strings grouped by functionality (users, calendar, etc.)
- **Easier Maintenance**: One file per feature makes updates intuitive

Migration Process

Step 1: Create Language Directory

Create a folder for your language in `src/languages/`:

```
src/languages/your_language/
```

Step 2: Create Split Language Files

Start with the most important controllers for your site. Here's the key mapping:

Essential Files (Start Here)

- **core.php** - Global strings used everywhere
- **login.php** - Login/authentication strings
- **home.php** - Homepage strings
- **users.php** - User management strings
- **calendar.php** - Calendar features

Complete Language File List

Controller File	Purpose	Typical Keys	Used By Controllers
core.php	Global strings	alert_, btn_, general	All controllers (auto-loaded)
about.php	About page	about_	about.php
absence.php	Absence management	abs_, absum_	absenceedit.php, absenceicon.php, absences.php, absum.php, remainder.php, statsabsence.php, statsabstype.php
alert.php	System alerts	alert_	alert.php (auto-loaded)
attachment.php	File attachments	attach_	attachments.php
bulkedit.php	Bulk editing	bulkedit_	bulkedit.php
calendar.php	Calendar features	cal_, caledit_	absenceedit.php, absum.php, bulkedit.php, calendaredit.php, calendarview.php, groupcalendaredit.php, month.php, remainder.php, year.php
calendaroptions.php	Calendar options	calopt_	calendaredit.php, calendaroptions.php, calendarview.php
config.php	Configuration	config_	config.php
database.php	Database management	db_	database.php
daynote.php	Daily notes	daynote_	daynote.php

Controller File	Purpose	Typical Keys	Used By Controllers
declination.php	Declinations	decl_	declination.php
email.php	Email system	email_	Various email functions
gdpr.php	Data privacy	gdpr_	dataprivacy.php, gdpr.php
group.php	Group management	group_	groupedit.php, groups.php
holiday.php	Holiday management	holiday_	holidayedit.php, holidays.php
import.php	Data import	import_	userimport.php
imprint.php	Legal imprint	imprint_	imprint.php
log.php	System logging	log_	log.php
login.php	Authentication	login_, password_	login.php, login2fa.php, logout.php, passwordreset.php, setup2fa.php, verify.php
maintenance.php	System maintenance	maint_	maintenance.php
message.php	User messaging	msg_	messageedit.php, messages.php
month.php	Monthly view	month_	month.php, monthedit.php
password.php	Password management	pwd_	passwordrequest.php, passwordreset.php, register.php, useradd.php, useredit.php

Controller File	Purpose	Typical Keys	Used By Controllers
pattern.php	Recurring patterns	pattern_	patternadd.php, patternedit.php, patterns.php
permission.php	User permissions	perm_	permissions.php
profile.php	User profiles	profile_	bulkedit.php, profile.php, useradd.php, useredit.php, viewprofile.php
region.php	Regional settings	region_	regionedit.php, regions.php
register.php	User registration	register_	register.php
remainder.php	Remainder tracking	remainder_	remainder.php, statsremainder.php
role.php	User roles	role_	roleedit.php, roles.php
statistics.php	Reports/analytics	stats_	statistics.php, statsabsence.php, statsabstype.php, statspresence.php, statsremainder.php
upload.php	File uploads	upload_	upload functions (auto-loaded)
user.php	User management	user_	useradd.php, useredit.php, userimport.php, users.php
year.php	Yearly view	year_	year.php

Step 3: Copy Keys from Legacy Files

1. Open your existing `english.app.php`

2. Find keys that start with the controller prefix (e.g., `abs_` for absence.php)
3. Copy them to the new controller file
4. Do this for all four legacy language files

Example for `absence.php`:

```php
<?php
/**
 * Absence Management Language File
 *
 * Contains all language strings for absence-related functionality
 * including absence types, editing, and reporting.
 */

// Absence list and management
$LANG['abs_list_title'] = 'Absence Types';
$LANG['abs_name'] = 'Name';
$LANG['abs_symbol'] = 'Symbol';
$LANG['abs_color'] = 'Color';

// Absence editing
$LANG['abs_edit_title'] = 'Edit Absence Type';
$LANG['abs_allowance'] = 'Allowance';
$LANG['abs_factor'] = 'Factor';

// Add all your custom abs_ keys here...
?>
```

Step 4: Test Your Migration

In `src/config/config.app.php`, set:

```php
define('USE_SPLIT_LANGUAGE_FILES', TRUE);
```

1. Visit different pages on your site
2. Look for missing strings or errors
3. Add missing keys to the appropriate controller files

Support and Resources

Getting Help

- **Check logs**: Look for PHP errors in your error logs
- **Test incrementally**: Migrate one controller at a time
- **Use fallback**: Keep old system active during migration
- **Ask for help**: Contact support if you encounter issues

Remember: Migration is optional and gradual. Your existing files continue to work, and you can migrate at your own pace while immediately benefiting from improved performance on migrated controllers.

Last updated: 2025-10-21 19:34