Datatypes: which ones the content uses and which ones break it

This guide teaches the datatype pages of the administration interface (Setup > Datatypes, /class/datatype/list, and one datatype, /class/datatype/view/): what a datatype is, where each one comes from, how much of the...

Source file: `doc/eng-us/guides/datatypes.md` Last changed 2026-11-04
https://doc.exponential.earth/exphelp/viewer/guides/datatypes.md

This guide teaches the datatype pages of the administration interface (Setup > Datatypes, `/class/datatype/list`, and one datatype, `/class/datatype/view/<identifier>`): what a datatype is, where each one comes from, how much of the content uses it, and how to find and repair content whose datatype nothing registers any more.

It is for administrators and developers who maintain the content model. The page is read only: it changes nothing. Every figure below was checked against the code (`kernel/private/classes/views/class/datatype.php` and `kernel/classes/expdatatypeinventory.php`) on the demonstration server on 3 November 2026.

Guides · Related: Class groups, The content model and editing content, All datatypes

In short

- A **datatype** stores and edits one kind of value of a class attribute: a text line (`ezstring`), an image (`ezimage`), a relation (`ezobjectrelationlist`). The kernel brings most of them; extensions add more (`eztags`, `ezgmaplocation` ...).
- The list shows every datatype `content.ini [DataTypeSettings] AvailableDataTypes[]` registers, grouped by who provides it: the kernel, then each extension. Each row says its PHP class and file, how many classes, class attributes, objects and object attributes use it, and its flags.
- A datatype the database uses but nothing registers any more is **Not registered**, in red, in a group of its own. The content that has it cannot be shown, edited or published until the datatype is back or the class is changed.
- One datatype's page shows its settings columns, its templates in this administration and in the default site, the classes and attributes that use it with their objects, all of its content (a page at a time, sortable), its tables and its help page.

Open it from Setup > Datatypes (the group "Content model", next to Classes) or with **Datatypes** on the class groups page (`/class/grouplist`). It needs the same access as the class pages (`class/*`).

1. The overview

Figure	What it counts
Registered	Datatypes `AvailableDataTypes[]` lists whose file was found and whose class could be created

Figure	What it counts
In use	Registered datatypes with at least one class attribute or object attribute
Not used	Registered datatypes no class and no object uses
Extensions providing datatypes	Extensions with at least one registered datatype
Class attributes	Attributes of the defined classes (version 0), every datatype
Object attributes	Attributes of the current version of every published object, one per language
Not registered	Datatypes in `ezcontentclass_attribute` or `ezcontentobject_attribute` that nothing registers (shown only when there are any)
Listed, file missing	Datatypes `AvailableDataTypes[]` lists but no datatype directory has (shown only when there are any)

The counts are two grouped queries, one over `ezcontentclass_attribute`, one over `ezcontentobject_attribute` joined to `ezcontentobject`. On a large database the second takes a few seconds (2.5 s on the demonstration server's 3 GB SQLite database with 200,000 object attributes), so both are kept for **five minutes** in one file of the cache directory. The header says when they were counted; **Count again** counts at once (`/(refresh)/1`).

2. Filters and search

- **Show**: All, In use, Not used, Not registered. They are view parameters: `/class/datatype/list/(filter)/unused`, `/(filter)/orphans`, `/(filter)/used`. "Not used" also lists the datatypes whose file is missing.
- **Provided by**: Everyone, Kernel, each extension, Not registered: `/(group)/kernel`, `/(group)/eztags`, `/(group)/orphans`. Both combine: `/class/datatype/list/(filter)/unused/(group)/kernel`.
- **Find a datatype** (with javascript): searches the identifier, the name, the PHP class, the extension and the file of the rows shown.
- **Sorting** (with javascript): a click on a column header sorts its table; numbers start with the largest, a second click reverses.

Without javascript the filters still work (they are links) and every row is shown in identifier order.

3. A row

Column	What it says
Datatype	The identifier (a link to its page), its name in the interface language and its state: Not registered, Listed, file missing, Not used
PHP class and file	The class it registered with `eZDataType::register()` and the file that defines it
Classes, Class attributes	Defined classes and their attributes of this datatype
Objects, Object attributes	Published objects (current version) with such an attribute, and those attributes (every language counts)
Flags	What the datatype says of itself

The flags:

Flag	From	Meaning
Translatable	`isTranslatable()`	Each translation has its own value
Searchable	`isIndexable()`	The search indexes its values
Collects information	`isInformationCollector()`	It can collect information from visitors (forms)
Sortable	`sortKeyType()`	Lists can be sorted by it (the tooltip names the sort key type)
Relations	`isRelationType()`	Its values point to other objects
From text	`isSimpleStringInsertionSupported()`	It can be filled from a plain string (import, REST)
From file	`isHTTPFileInsertionSupported()`, `isRegularFileInsertionSupported()`	It can be filled from a file
Packages	`serialize_supported`	Its class attribute settings go into packages

4. One datatype

`/class/datatype/view/<identifier>`, for example `/class/datatype/view/ezimage`:

- **About this datatype**: identifier, name, provider (the kernel or the extension), PHP class, file, whether `AvailableDataTypes[]` lists it, the flags, the columns of its value (`data_text`, `data_int`, `sort_key_*` ...), the columns of its class attribute settings (`data_int1` ... `data_text5`) and its tables (`ezimagefile` for `ezimage`). The columns and tables are found by reading the datatype's code, the same heuristic as the generated datatype pages: a guide, not a promise.
- **Templates**: `content/datatype/view`, `edit`, `collect`, `result/info` and `class/datatype/view`, `edit` for the datatype, each with the file the design chain finds first, for this administration and for the default siteaccess (`site.ini [SiteSettings] DefaultAccess`). Override templates matched by class or attribute are not applied here; they win over these.
- **Classes using it**: each class (a link to class/view) with its attributes of this datatype, its objects and object attributes.
- **Class attributes**: each attribute with its class, its flags (Required, Searchable, Translatable, Collects information) and its settings columns that are set (`data_int1=600` ...), with its objects and object attributes.
- **Content**: every object of those classes that you may read, once each (its main node), with its class, section, languages, owner and dates. Sort it by modified (the default, newest first), published, name or class with the column heads (a second click reverses the order), show 10, 25, 50 or 100 a page and narrow it to one class. The choice stays in the address, for example `/class/datatype/view/ezimage/(sort)/name/(order)/asc/(limit)/50/(class)/5/(offset)/50`; the count is kept five minutes.
- **Help page**: the generated page of the datatype in the help center, where there is one.

5. Not registered: repair content whose datatype is gone

A class attribute keeps its datatype's identifier (`data_type_string`) in the database. When an extension that provided a datatype is deactivated or removed, or its `content.ini.append.php` no longer lists it, the identifier stays and the kernel cannot create the datatype: the objects of the class fail to show, edit or publish.

1. Open `/class/datatype/list/(filter)/orphans`. The notice says how many class and object attributes have such a datatype.
2. Open the datatype. **Classes using it** names the classes; **PHP class** and **File** are empty.
3. Either activate the extension that provides it (Setup > Extensions) and check that its `settings/content.ini.append.php` adds it to `[DataTypeSettings] AvailableDataTypes[]` and its directory to `ExtensionDirectories[]`, then clear the INI cache; or edit each class and remove the attribute (its values are removed with it) or move the content to a class without it.
4. **Count again**: the row is gone from Not registered.

"Listed, file missing" is the other way round: a setting lists a datatype whose file no datatype directory has. Nothing uses it; remove it from `AvailableDataTypes[]` or activate its extension.

6. Databases

The counts run on every database the kernel supports. On SQLite, MySQL, PostgreSQL and Oracle they are the same two `GROUP BY data_type_string` statements through `eZDB::arrayQuery()` (the column names are read in any case, as Oracle returns them in upper case). On MongoDB they are aggregation pipelines (`$group` with `$addToSet`; the object attributes with a `$lookup` of `ezcontentobject`), since the driver's SQL translation answers only single-table SELECTs without `GROUP BY`.

References

- Module and view: `kernel/class/module.php` (view `datatype`, parameters `Action`, `DataTypeString`), `kernel/class/datatype.php`, `kernel/private/classes/views/class/datatype.php`.
- Data: `kernel/classes/expdatatypeinventory.php` (`registry()`, `usage()`, `build()`, `filter()`, `classAttributes()`, `templates()`, `readSource()`); cache file `class-datatype-*.json` in the cache directory, five minutes.
- Templates: `design/admin4/templates/class/datatype_list.tpl`, `datatype_view.tpl`, `datatype_flags.tpl` (the same files in `design/admin/templates/class/`); script `javascript/page/class_datatypes.js`; the sort buttons in `stylesheets/modules/structure.css` (`.a4-class-datatypes-sort`).
- Menu: `settings/menu.ini` `Links[datatypes]`, `GroupLinks_content_model[]`; context help `settings/exphelp.ini` `[ContextHelp] Map[class/datatype]`.
- Tests: `tests/tests/kernel/classes/expDataTypeInventoryTest.php` (no database), `tests/tests/kernel/classes/expDataTypeInventorySqliteTest.php` (SQLite in memory).

Become an Exponential Partner

Partners share their information, provide funding, contribute back and develop the products with us, and that is how Exponential grows. A partner website and dashboard are planned for mid-2027: please become a partner today to speed up our growth rate and land new features faster.

Partners: start here | How to apply

Exponential Copyright © 1998-2026 Exponential Foundation and 7x
Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.3 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled "GNU Free Documentation License".
This documentation is free documentation, licensed under the GNU Free Documentation License 1.3 or later:
https://www.gnu.org/licenses/fdl-1.3.html.