Dataverse Field Types with Naming Standards and Considerations

Below is a comprehensive table that outlines the Dataverse field types with their OOB and custom naming examples / conventions. It also expands on naming standards, casing (e.g., camelCase vs. snake_case), and other considerations for each field type. Following the table is an expanded section on relationship and lookup naming best practices.

Field Type	OOB Schema Column Naming Example	Custom Field Naming Convention Example	Description & Purpose	Naming S Considera
Single Line of Text	firstname,lastname,name	new_firstNameor new_shortTextField	Stores brief textual data such as names, titles, or identifiers.	OOB fields case and c fields shou publisher p followed by Many team (e.g.,new_ snake_cas
Multiple Lines of Text	description,comments	new_longTextField	Provides a text area for longer, free-form inputs like detailed descriptions or notes.	Use clear, all lower-ca custom fie prefix and case (came for readabi
Option Set (Choice)	statuscode,preferredcontactmethodcode	new_statusOption	Presents a dropdown list where the user selects a single value from a predefined list.	OOB name “code” to i Custom fie publisher p consistent or camelCa the field’s p
Multi-Select Option Set	(Rarely available OOB)	new_multiSelectField	Enables users to select multiple options from a predefined list of values.	Follow sim Option Set include the use a cons
Whole Number	numberofemployees	new_wholeNumberField	Stores integer values with no decimals, typically for counts, quantities, or rankings.	OOB fields with no sep should add and use ca snake_cas (e.g.,new_ enhance c
Decimal Number	exchangeRate	new_decimalField	Captures numbers with defined precision and scale—ideal for exact numeric measurements.	Names sho precision; concise, w the publish consistent (camelCas

Field Type	OOB Schema Column Naming Example	Custom Field Naming Convention Example	Description & Purpose	Naming S Considera
Floating Point Number	`rating`	`new_floatField`	Stores approximate numeric values where exact precision isn’t essential (e.g., ratings, scientific data).	Naming sh approxima OOB fields case; custo publisher p consistent
Currency	`revenue`,`budget`	`new_currencyField`	Designed for monetary values, complete with built-in currency formatting and conversion support.	Use short, OOB fields while custo publisher p consistent (e.g.,`new_`
Date and Time	`createdon`,`modifiedon`	`new_dateTimeField`	Records dates and times, configurable as “date only” or “date and time” (with or without time zone awareness).	Names sho temporal n fields are t Custom fie publisher p camelCase (e.g.,`new_` clarity.
Boolean (Two Options)	`isactive`,`emailoptout`	`new_isEnabled`or `new_booleanField`	Represents binary choices (Yes/No, True/False) for flags or toggles.	Typically, b “is”, “has”, denote a c are in lowe include the follow the c consistentl
Lookup	`ownerid`,`parentcustomerid`	`new_relatedEntityId`	Creates a relationship by referencing another entity record.	Should cle target entit OOB fields custom fie publisher p camelCase (e.g.,`new_` for clarity.
Customer	`customerid`	`new_customerField`	A specialized lookup that can reference either an Account or a Contact, offering flexibility for customer-related data.	Use concis its dual-ref naming is s custom fie publisher p descriptive (e.g.,`new_`
Owner	`ownerid`	(System-managed; custom owner fields are rare)	Indicates the record owner (user or team) for	Naming is consistent

Field Type	OOB Schema Column Naming Example	Custom Field Naming Convention Example	Description & Purpose	Naming S Considera
			security and access control purposes.	customizat uncommon
Unique Identifier (GUID)	`accountid`,`contactid`	(System-managed; custom GUID fields are uncommon)	A system-generated globally unique identifier serving as the primary key for records.	Names foll entity nam lower-case customizal by the syst
Image / File	`entityimage`	`new_imageField`	Stores binary data like images or files, often used for profile pictures or logos.	Use simple OOB fields word; cust publisher p formatted casing star snake_cas
Calculated Field	`fullname`(derived from other fields)	`new_calculatedField`	Dynamically computes its value based on expressions or formulas referencing other fields.	Naming sh field is deri like "calcul OOB fields naming; cu publisher p case forma
Rollup Field	(Defined via system relationships/aggregations)	`new_rollupField`	Aggregates data from related records (for example, summing values or counting related items).	Names sho aggregativ purpose. C include the follow cam conventior calculated

Expanded Naming Best Practices for Relationships and Lookups

Relationship Naming

- **Clarity & Descriptiveness:**

  Relationship names should clearly indicate the nature of the association. A common convention is to combine the primary entity name with a descriptive term for the relationship (e.g., `account_contact` for an account's primary contact).

- **Publisher Prefix for Custom Relationships:**

  For custom relationships, prepend the publisher's unique prefix (e.g., `new_`) to avoid collisions with OOB relationships.

- **Case Consistency:**

  Use a consistent case—whether lower-case, camelCase, or snake_case—across all relationship names to ensure uniformity throughout the solution.

- **Avoid Ambiguity:**

  Choose names that precisely communicate the relationship's purpose rather than relying on abbreviations. This enhances readability and maintainability.

Lookup Field Naming

- **Entity Reference in the Name:**

  Lookup field names should incorporate the name of the referenced entity and typically end with "id" (e.g., `primarycontactid` or `new_primaryContactId`).

- **Descriptive Purpose:**

  The lookup name should reflect its role (for example, differentiating between "manager" and "supervisor" if multiple lookups exist on a single entity).

- **Consistency Across the Model:**

  Whether using camelCase or snake_case, keep the naming convention consistent. OOB lookups are usually lower-case and custom fields should follow suit, starting with the publisher prefix.

- **Relationship Direction:**

  In polymorphic scenarios or when multiple lookups exist, consider including additional descriptive text to clarify the relationship (e.g., `new_parentAccountId` vs. `new_primaryAccountId`).