Data Types

Tuesday, September 9, 2025 9:23 AM

Data types

Primitive data types are the most common data types that you encounter. The primitive data types are: `anytype`, `boolean`, `date`, `enum`, `guid`, `int`, `int64`, `real`, `str`,`timeOfday`, and `utcdatetime`.

- Date - This data type contains the day, month, and year. You can write out the date as literals by using the format day\month\year. The year must be entered as four digits. The dates must be within the January 1, 1900 and December 31, 2154 range. We recommend that you use `utcDateTime` instead of `Date` because it combines `date`, `timeOfDay`, and time zone information into a single data type.
- Enumerations (enums) - Are a named list of literals. For example, there's an enum called `NoYes`, with a list of literals No and Yes. Each literal can also be represented by a number. The first literal has the number 0, the next literal has the number 1, and so on. You can also provide your own values in the enum metadata. To reference an enum value, you would use the enum name followed by two colons and the name of the literal. Considering the `NoYes` enum again, to reference the Yes literal, you would write `NoYes::Yes`.
- GUID - This data type is a globally unique identifier (GUID) value. A `GUID` number is unique across all computers and networks.
- Int - These values are 32-bit wide integer signed numbers that don't include a decimal place. There's no support for unsigned integer types.
- Int64 - A larger, signed integer data type that works like the `int` data type, except that it's 64 bits wide compared to 32 bits. The `int64` type has a greater range for larger numbers.
- Real - These data types hold number values with decimals.
- Str - This data type contains values that are a string of characters. The `str` value must be wrapped in double quotation marks ("text") or single quotation marks ('text').
- TimeOfday - This data type is an integer value that represents the number of seconds that passed since midnight.
- Utcdatetime - This data type is a combination of the `date` type and the `timeOfday` data type. The format is yyyy-mm-ddThh:mm:ss with the default value being 1900-01-01T00:00:00.

Composite data types

Along with primitive data types, there are composite data types. Composite data types include arrays, containers, classes, delegates, and tables.

- Arrays - Contain a list of items that have the same data type. You can retrieve an element in the array with an integer index. You can use arrays in different ways. Dynamic arrays are open and fixed-length arrays specify a certain length. Partly on disk arrays can be dynamic or fixed, with an option to determine how many items are held in memory.
- Containers - Are dynamic collections of primitive items or other containers. Containers can be useful to pass different types of values, but they aren't ideal when used in processes that frequently change the size or contents.

- Classes - Can be used as a data type to declare an instance of a class within a method.
- Delegates - Can be defined on a method, table, form, or query. When a delegate is called, it also calls their subscriber methods. Delegates never return a value and have a default value of *no handlers added*, which means nothing is called when you call them.
- Tables - Can be used to declare an instance of a table. Once a table instance is declared, you can manipulate the data within the table. We recommend that the table variable has the same name as the table, but you should use camel casing. For example, the table EcoResProduct would be declared by using camel casing as ecoResProduct.

From <https://learn.microsoft.com/en-us/training/modules/get-started-xpp-finance-operations/2-base-types>

Variables

Variables are identifiers that designate a storage location where a value of a data type is stored. The size, precision, default value, and conversion functions depend on the variable's data type. Variables can be declared anywhere in a code block in a method; they don't have to be declared at the beginning of a method.
In a class, member (global) variables and local variables have different scopes. Global variables can be used throughout the instance (excluding static methods) of the class, while local variables are only available within the scope of the class method.
When you declare a variable, you must assign the data type and then assign the variable name. At this point, you can assign a value or leave it open to set the value. The following procedure is an example of declaring a variable.

1. Declare the data type inline and within code. In this example, use str for string data type.
2. Name the variable. It must start with a letter or underscore. We recommend using meaningful names for variables that follow camel case. Camel case is where you don't capitalize the first letter of the name, but each word or abbreviation that follows the first in the phrase is capitalized (for example, purchaseOrderLine). Spacing or punctuation isn't permitted.
3. If you're assigning a value, add the equal sign and the value of the variable. String values must be either surrounded by double quotation marks or by single quotation marks for hard-coded text. It's best practice to use labels in strings, excluding the few exceptions where this isn't feasible. Single quotation marks around hard-coded text suppress the best practice error that you would get for not using a label.
4. End the declaration with a semicolon.

From <https://learn.microsoft.com/en-us/training/modules/get-started-xpp-finance-operations/2-base-types>