

[Back to Reference](#)

Contents

- [1 Prerequisites](#)
- [2 Schema](#)
- [3 Statements](#)
- [4 Sites](#)
- [5 Intrinsic Functions](#)
- [6 Intrinsic Variables](#)

Note Pages marked with ** have not completed the review and editorial process and may be incomplete.

Prerequisites

[Understanding Bungee Logic](#)

Schema

Bungee Logic is built from some simple building blocks that should look familiar:

- [Class](#) ** — a structure that combines multiple pieces of data into a single object
- [Field](#) ** — an encapsulation of data within a class or object
- [Function \(\)](#) ** — contains **Bungee Logic** and is used to implement specific program functionality
- [Argument](#) ** — a parameter passed to to or from a function
- [Var](#) ** — a variable declaration

Statements

Statements add the logic and control flow to classes:

- [OnError](#) ** — similar to the try/catch statement in other languages (**This is currently not implemented**)
- [Assignment](#) ** — assigns a value to a member of an object
- [Call Function \(\)](#) ** — calls a user or system-defined function
- [Case \(\)](#) ** — conditionally executes a block of code
- [Collection Iteration](#) ** — a loop that executes (at most) once for each element of a collection
- [Comment](#) ** — a one line comment
- [Dialog Message](#) ** — a standard dialog for notifying the user, obtaining user input, or debugging
- [For Iteration](#) ** — a counting iteration
- [If Condition](#) ** — evaluates a series of one or more cases, similar to a switch statement
- [Log Message](#) ** — log information to the event queue, useful for debugging
- [Open Dialog](#) ** — use a custom designed form for user interaction

Bungee_Logic

- Return ** — immediately returns execution to the caller from the current function
- While Iteration ** — continues iterating as long as the specified expression is satisfied

Sites

A Site is a Bungee term that simply specifies what type of object you will be using. The site type basically specifies scope of the object to select.

In the Bungee Logic Assignment statement for example, the left and right values are specified by sites.

Important: Not all sites are valid in every context. For example, a site on a Form control cannot use the Var site as it is not executing in the context of function scope, but at class scope, so Path should be used.

- Path **
- Var **
- Data **
- Object **
- Expression **
- Adapter **
- Collection **
- Type **
- SysGlobal **
- AppGlobal **
- Bag **
- Datasource **

Intrinsic Functions

An intrinsic function is a function that is always available whether it is used or not.

Every field has two intrinsic functions.

- OnInit ** — This function is executed when the field is first created at runtime. Similiar to a C++ constructor.
- OnChange ** —This function is executed when the value of the field changes at runtime.

Every variable within a connection has two intrinsic functions.

- Model **
- View **

Intrinsic Variables

Intrinsic Variables are variables that are automatically created inside various Bungee Logic statements to simplify logic programming. These variables should be considered system variables and should not be deleted or renamed - this will result in undefined behavior.

Bungee_Logic

Note You can however, change the type (see the Nested Iteration section of the Collection Iteration page for more details).

Intrinsic variables often are automatically initialized with important information and can be modified to change the behavior of statements. Intrinsic variables are used by Iteration statements and OnError statements, among others. The following intrinsic variables are available in Bungee Connect:

- error ** — Implemented in the Call Function () statement (when the Handle on Error property is selected) and the OnError statement to store the error value.
- oldValue ** — Used in the OnChange intrinsic function to store the previous value of a field.
- StopIterating** — Setting this to *true* will terminate the loop after the current iteration finishes in the For Iteration loop, the Collection Iteration, and the While Iteration.
- Length** — Used by the Collection Iteration statement, this contains the number of elements in the collection.
- CurrentIndex** — Used by both the Collection Iteration and the For Iteration statements, this contains the current value of the iteration counter.
- CurrentElement** — Used by the Collection Iteration statement, this references the element at the current index of the iteration. The type of this variable can be safely changed if required.