Options Merging

The framework merges a component's defaults with the options specified by the user at runtime. Default options for a component are registered by the component developer using the `fluid.defaults()` function. The framework combines these values with those supplied by the user to the component creator function.

On this Page

Left unchecked, this options merging code becomes unruly and verbose. The vast majority of merging can be taken care of by a simple call to a standard "object merging function" such as `jQuery.extend` - however, there are a few cases where this behaviour is inappropriate, and where manual merging code might still be required.

The Fluid component API therefore contains a specialised merging routine, `fluid.merge`, which allows the user to specify an (optional) more fine-grained *merge policy object*, which allows detailed behaviour during options merging to still be specified by a declarative strategy.

`fluid.merge` signature

The signature to `fluid.merge` is

```
fluid.merge(policy, target, source-1, ... source-n);
```

where `policy` is the *merge policy object* (may be empty), `target` is the object to (destructively) be the target of the merge operation, and `source-1` through `source-n` are a list of a number of "source" options structure from which the merge is to be performed.

Use of `fluid.merge`

The framework user should never need to invoke fluid.merge directly - it is invoked automatically by the framework as part of standard component initialisation. However, the component author does have the capability of directing the operation of fluid.merge by means of specifying the policy object. Every standard Fluid component accepts a top-level option named `mergePolicy` which will be supplied as the `policy` argument to fluid.merge. This option itself also undergoes merging (although the user may not specify policy for the merging of the `mergePolicy`!) and users may contribute material into the `mergePolicy` from any parent grades of the component, arguments, etc.

Structure of the Merge Policy Object

The merge policy object is a hash of keys, which represent *EL paths* into the target object, onto values which represent a policy.

By default if there is no policy specified, a deep merge is done together with expansion. Firstly, any IoC references and expanders in the source objects will be expanded. Secondly, everything in the source objects are copied over the target object, in a manner very similar to the operation of the jQuery API method `$.extend(true, ...)`. Anything that existed in the target but was not present in any of the source objects will continue to be present in the resulting merged object.

The following are the policy types that are supported, determined by the key's value in the policy object. For the custom policies in the first three rows, combinations may be applied by separating them with commas - e.g. `"noexpand, nomerge"`.

Policy value	Description
`"replace"`	If any value is found at the EL path in a "source" object, the value held at the "target" path will be cleared (that is, blasted back to an empty state with no properties) before the values are copied.
`"noexpand"`	Apply no expansion to any of the options material in the tree below this path. Unless you also specify "nomerge", although no expansion occurs, the contents of the source options will be merged together as if they consist of pure JSON material.
`"nomerge"`	Can be specified together with, or separately from, "noexpand". This directs the framework to perform no merging when multiple values are discovered at the same path - instead, the rightmost value from any of the sources will be chosen as the final value. The framework will make a good attempt to preserve the exact object handle present at that path in the final options structure. This is the appropriate policy when the option value does not consist of pure JSON material but instead contains some form of "exotic material" such as a Fluid component or native object (e.g. Date, Float32Array, Socket, etc.). The framework detects some kinds of native object (e.g. DOM nodes, jQuery objects) and protects them from merging without the use of this option.
Any other string value	The value will be interpreted as an EL path into a source object. If no user-supplied value is present at this options path (other than those supplied in defaults), the value will be taken from the final merged value held at this specified EL path. That is, this style of policy is used to specify that an options value has a default value which defaults to the value of *another supplied option*.

function	This option allows the user to take complete control of the merging process at this path with a custom *policy function*. The value holds as a function which accepts two arguments (target, source) representing the sub-objects which are about to enter merging. The function will be invoked on these arguments and no further operations will be performed on this subtree. The return value of the function will become the next "running value" of the target and may be supplied as argument 1 to another invocation of the same policy function. These semantics are the same as those of the standard functional algorithm known as reduce (or fold, accumulate, etc.)