Rocket software

Rocket Uniface Library 10.4

Passing Struct Parameters

Structs can be passed as parameters by reference or by value. The default behavior is to pass Struct parameters by reference in partner operations and functions, and to pass them by value in public operations, but it is possible to override this behavior when declaring the parameters.

The content of a `struct` parameter is not a Struct, but zero or more references to Struct nodes, just as it is for `struct` variables. This is in contrast to scalar data types such as `string`. References are memory pointers so it is only possible to pass them between ProcScript modules that share the same memory. This is always the case for partner operations and functions, but may not be the case for public operations.

For this reason, the default behavior is to pass Struct parameters by reference in partner operations and functions, and by value in public operations. When passed by reference, the parameter passes a copy of the reference to the called function or operation, so an additional reference to the same struct is created. When passed by value (the default for public operations), a copy of the Struct is made and this can have consequences when the data is returned.

It is possible to override the default behavior by explicitly defining how Struct parameters are passed using the `byRef` or `byVal` qualifiers. This can be useful if you know that the calling and receiving component instances are running in the same process.

If neither the `byRef` nor the `byVal` qualifiers are used, the default behavior depends on whether the parameter is declared in a public operation or not. However, because of the functional difference between the two, it is advisable to explicitly specify the intended qualifier.

If an operation with a `byref` struct parameter is activated in a component running in a different process, this will result in the following error:

```
-73 <ACTERR_REMOTE_NOT_SUPPORTED> "Operation with byref-Struct parameter cannot be activated across processes"
```

Passing Structs by Reference

Structs are passed by reference when Struct parameters are declared in a partner operation or function, or when the `byRef` qualifier is used in any `params` declaration. For example:

```
public operation MyOperation1
params
 byRef struct myRefParam : INOUT
endparams
...
end; end MyOperation1
```

Passing Structs by reference is more efficient than passing them by value, and therefore provides better performance. There are also no synchronization problems because the parameters refer to the same Struct.

For this reason, it can be very useful to pass Struct parameters by reference whenever possible, even in public operations. You can only do this if you know that the calling and receiving component instances are running in the same process. For example, you may have multiple instances of the same component running at the same time which need to have access to the same Struct data.

However, trying to remotely execute an operation that has `byRef` parameters returns error `-73`.

Passing Structs by Value

Structs are passed by value when Struct parameters are declared in a public operation, or when the `byVal` qualifier is used in the declaration. For example:

```
operation MyOperation2
params
 byVal struct myValParam : INOUT
endparams
...
end; end MyOperation2
```

Passing Structs by value is the only way to exchange information between components that are running in different processes, but it is fundamentally different from passing Structs by reference, because a copy of the Struct is made. If the Struct data is both sent and returned, you need to consider what happens if multiple Struct variables point to the same Struct. For example, assume the following:

- Struct variable `vBook` references a complex Struct
- Struct variable `vChapter` references a member of that Struct
- Operation `Annotate` adds a specific annotation (`$tags->translate`) to each of the `vBook` Struct's members.
- Operation `Annotate` has an INOUT Struct parameter `pMyStruct`

If Struct `vBook` is passed to operation `Annotate` by reference, all the members of `vBook` will have get the `translate` annotation. This annotation will also be visible from Struct variable `vChapter`, because it points to one of the members of that Struct.

If Struct `vBook` is passed to operation `Annotate` by value, the effect is very different—after the operation returns, variable `vBook` refers a whole new Struct. The members of this Struct all have the `translate` annotation, but `vChapter` still refers to the member of the Struct to which variable `vBook` referred before the operation call. That member would be unchanged.

Passing Structs by value can also degrade performance if the Structs being exchanged are very large.

Errors

At runtime, the following errors may be returned in `$procerror` after activating an operation with Struct parameters.

Table: Errors Commonly Returned in `$procerror` after Passing Struct Parameters

Value of $procerror	Error Constant	Meaning
-73	ACTERR_REMOTE_NOT_SUPPORTED	Operation with byref-Struct parameter cannot be activated across processes.
-1576	DESERIALIZEERR_INTERNAL	Internal error. Indicates that the data may have been corrupted when passed by value.
-1579	DESERIALIZEERR_HANDSHAKE	The sending and receiving components are running in different versions of Uniface.

Related concepts