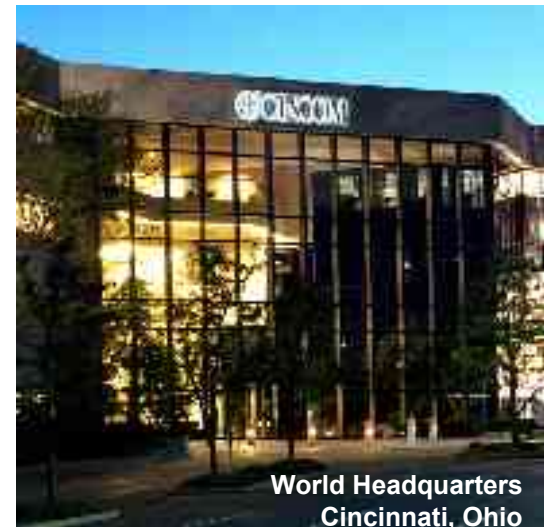




Moving to ObjectStudio 8

Mark Grinnell, Lead Software Engineer

Andreas Hiltner, Lead Software Engineer



World Headquarters
Cincinnati, Ohio

SIMPLIFICATION THROUGH INNOVATION™

Welcome
December 13, 2006

The Goal of the OS8 project was to provide complete code compatibility with existing ObjectStudio applications.

Unfortunately this is not always possible.

Here we describe what the differences are and provide guidelines to write compatible code.

Most of this can be applied to existing code for an easier transition to OS8.

Syntax differences

- Unary minus
- Methods and Method Params
- Global Variables
- Class and Instance Variables
- Literals / Immutables
- Namespaces
- LiteralBindingReference

Unary minus

- ObjectStudio8 does not support unary minus as a true message send.

ObjectStudio classic allows

– 500 *“please note the space between ‘-’ and ‘500’”*

Methods and Method Params

For backward compatibility we allow the overwrite of method parameters, e.g.

```
methodA: param
```

```
    param := param asString
```

```
    ^param
```

Methods and Method Params cont.

VisualWorks does not allow that.

```
methodA: param
```

```
  | methodTemp |
```

```
  methodTemp := param asString
```

```
  ^methodTemp.
```

Global Variables

Global variables do not exist in ObjectStudio8, but are now shared variables, and are scoped by a context.

The familiar ObjectStudio globals are defined in the ObjectStudio.Globals namespace, and should be declared and initialized there.

Class and Instance Variables

The compiler in ObjectStudio 8 does not allow duplicate class variables or instance variables, and raise an exception during the compilation.

Literals / Immutables

Literals cannot be changed at runtime.

Literal arrays, often used in `Controller>>createItems`, cannot be changed and throw an exception if you attempt to do so.

To avoid that, simply add 'copy' to the specific array or string.

Literals / Immutables

For example, this throws an exception

```
a := {#a #b}.
```

```
a at: 1 put: #c.
```

but this does not, and is correct:

```
a := {#a #b} copy.
```

```
a at: 1 put #c.
```

Namespaces

Namespaces allow VisualWorks to be very flexible in how it handles add-in components from a multiplicity of vendors.

ObjectStudio 8 uses the same mechanism to separate the original VisualWorks classes from the current ObjectStudio classes. The namespace 'ObjectStudio' is used for that.

All loaded classes will end up in that namespace.

LiteralBindingReference

LiteralBindingReference represents a binding reference that appears in a method's source code literally, using `# {Namespace.Class} syntax`

It can be used to query whether a particular variable has been defined or class has been loaded, e.g.

```
# {LoadableApplications} isDefined  
  
ifTrue: [ self doSomething ]  
  
ifFalse: [ self loadSomething ].
```

Class Library Differences

ObjectStudio8 shares Collection and Magnitude classes with VisualWorks.

This leads to some differences we describe in the following slides.

Equality of Objects in Collections

Equality tests are different in certain cases between common VW and OS objects.

The equality test is determined by the receiver and at the same time by the context in which it is sent.

```
OS8: (Set with: {2006 12 3} asDate)  
includes: {2006 12 3} -> false
```

```
OS7: (Set with: {2006 12 3} asDate)  
includes: {2006 12 3} -> true
```

Subclassing Collections

If you create a subclass of a collection class, you need to implement the instance method `copyEmpty`: in order to copy any new instance variables you'd like to preserve.

Collection return values

Collections in classic ObjectStudio used to return the collection itself and ObjectStudio8 still supports that.

However, if you want to write compatible code of enhance an existing VW class, you need to be aware that `#at:put:` returns the put-value.

In case you need the collection itself, add the method 'yourself' at the end, e.g.

```
coll := coll at: 1 put: #a; yourself.
```

BlockClosures

The following methods differ between classic ObjectStudio and OS8.

whileTrue:

whileTrue: always returns nil in OS8, instead of false as it does in classic ObjectStudio.

whileFalse:

whileFalse: always returns nil in OS8, instead of true as it does in classic ObjectStudio.

Date

The new class Date only allows valid dates.

Wrong dates cause an exception to be raised.

Note that month names are case insensitive in OS8.

Uppercase month returned `nil` in classic ObjectStudio

Some methods behave differently. A complete list can be found in the documentation.

Process/Thread Management

The VisualWorks virtual machine is not designed to use native operating system threads, as does the classic ObjectStudio VM.

Instead, a Smalltalk scheduler schedules all Smalltalk processes. You can still use the class Thread, but these are no longer native threads.

The ThreadBrowser is replaced by the VW ProcessMonitor.

`#fork/#forkActive` still work

Process/Thread Management cont.

The benefits are:

- OS GUI events are not sent to the wrong thread
- Unbreakable deadlocks do not occur
- Smalltalk processes are very cheap to start and generate very little overhead.

On the other hand, machines with multiple CPUs cannot spawn those threads, and the process stays on the CPU it was started on.

SendHooks

SendHooks don't exist any more.

They got replaced by method wrappers.

Breakpoints don't have this big overhead any more.

User-written Primitives

ObjectStudio 8 is compiled as C++.

This may require you to make some changes to the existing code.

The C++ compiler does not handle implicit casts, so some changes have to be applied here.

Recompile all DLLs using the same compiler flags as ObjectStudio.

An example of a makefile is provided.

User-written Primitives cont.

Pointer to Smalltalk objects are implemented as instances of class OPTR.

You must use the provided macros to access data within objects.

Nil is now a special instance of class OPTR. Any pointer should be checked against Nil, not NULL or 0 (zero).

Questions?



**© 2005 Cincom Systems, Inc.
All Rights Reserved
Developed in the U.S.A.**

CINCOM and the Quadrant Logo are registered trademarks of Cincom Systems, Inc.
All other trademarks belong to their respective companies.