clCreateCommandQueue

Create a command-queue on a specific device.

```
cl_command_queue
clCreateCommandQueue(     cl_context context,
                          cl_device_id device,
                          cl_command_queue_properties properties,
                          cl_int *errcode_ret)
```

Parameters

context

Must be a valid OpenCL context.

device

Must be a device associated with *context*. It can either be in the list of devices specified when *context* is created using clCreateContext or have the same device type as the device type specified when the *context* is created using clCreateContextFromType.

properties

Specifies a list of properties for the command-queue. This is a bit-field. Only command-queue properties specified in the table below can be set in *properties*; otherwise the value specified in *properties* is considered to be not valid.

Command-Queue Properties	Description
CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE	Determines whether the commands queued in the command-queue are executed in-order or out-of-order. If set, the commands in the command-queue are executed out-of-order. Otherwise, commands are executed in-order.
CL_QUEUE_PROFILING_ENABLE	Enable or disable profiling of commands in the command-queue. If set, the profiling of commands is enabled. Otherwise profiling of commands is disabled. See clGetEventProfilingInfo for more information.

errcode_ret

Returns an appropriate error code. If *errcode_ret* is NULL, no error code is returned.

Notes

The OpenCL functions that are submitted to a command-queue are enqueued in the order the calls are made but can be configured to execute in-order or out-of-order. The *properties* argument in **clCreateCommandQueue** can be used to specify the execution order.

If the CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE property of a command-queue is not set, the commands enqueued to a command-queue execute in order. For example, if an application calls clEnqueueNDRangeKernel to execute kernel A followed by a clEnqueueNDRangeKernel to execute kernel B, the application can assume that kernel A finishes first and then kernel B is executed. If the memory objects output by kernel A are inputs to kernel B then kernel B will see the correct data in memory objects produced by execution of kernel A. If the CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE property of a commandqueue is set, then there is no guarantee that kernel A will finish before kernel B starts execution.

Applications can configure the commands enqueued to a command-queue to execute out-of-order by setting the CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE property of the command-queue. This can be specified when the command-queue is created or can be changed dynamically using **clCreateCommandQueue**. In out-of-order execution mode there is no guarantee that the enqueued commands will finish execution in the order they were queued. As there is no guarantee that kernels will be executed in order, i.e. based on when the clEnqueueNDRangeKernel calls are made within a command-queue, it is therefore possible that an earlier clEnqueueNDRangeKernel call to execute kernel A identified by event A may execute and/or finish later than a clEnqueueNDRangeKernel call to execute kernel B which was called by the application at a later point in time. To guarantee a specific order of execution of kernels, a wait on a particular event (in this case event A) can be used. The wait for event A can be specified in the event_wait_list argument to clEnqueueNDRangeKernel for kernel B.

In addition, a wait for events or a barrier command can be enqueued to the command-queue. The wait for events command ensures that previously enqueued commands identified by the list of events to wait for have finished before the next batch of commands is executed. The barrier command ensures that all previously enqueued commands in a command-queue have finished execution before the next batch of commands is executed.

Similarly, commands to read, write, copy or map memory objects that are enqueued after clEnqueueNDRangeKernel, clEnqueueTask or clEnqueueNativeKernel commands are not guaranteed to wait for kernels scheduled for execution to have completed (if the CL_QUEUE_OUT_OF_ORDER_EXEC_MODE_ENABLE property is set). To ensure correct ordering of commands, the event object returned by

clEnqueueNDRangeKernel, clEnqueueTask or clEnqueueNativeKernel can be used to enqueue a wait for event or a barrier command can be enqueued that must complete before reads or writes to the memory object(s) occur.

Errors

clCreateCommandQueue returns a valid non-zero command-queue and *errcode_ret* is set to CL_SUCCESS if the command-queue is created successfully. Otherwise, it returns a NULL value with one of the following error values returned in *errcode_ret*:

- CL_INVALID_CONTEXT if *context* is not a valid context.
- CL_INVALID_DEVICE if *device* is not a valid device or is not associated with *context*.
- CL_INVALID_VALUE if values specified in *properties* are not valid.
- CL_INVALID_QUEUE_PROPERTIES if values specified in *properties* are valid but are not supported by the device.
- CL_OUT_OF_HOST_MEMORY if there is a failure to allocate resources required by the OpenCL implementation on the host.

Specification

OpenCL Specification

Also see

clGetCommandQueueInfo, clReleaseCommandQueue, clRetainCommandQueue, clSetCommandQueueProperty, clCreateContext, clCreateContextFromType