Class AppDriver

Table of contents

- Defined in File app_driver.hpp

Nested Relationships

Nested Types

- Struct AppDriver::DriverMessage

Class Documentation

class AppDriver

Public Types

enum class AppStatus

Values:

enumerator kNotInitialized

enumerator kNotStarted

enumerator kRunning

enumerator kFinished

enumerator kError

enum class DriverMessageCode

Values:

enumerator kCheckFragmentSchedule

enumerator kWorkerExecutionFinished

enumerator kTerminateServer

Public Functions

explicit AppDriver(Application *app)

virtual ~AppDriver()

void run()

std::future<void> run_async()

CLIOptions *options()

AppStatus status()

Note that the application status is not updated when the application is running locally.

FragmentScheduler *fragment_scheduler()

MultipleFragmentsPortMap *all_fragment_port_map()

void submit_message(DriverMessage &&message)

void process_message_queue()

Public Static Functions

static bool get_bool_env_var(const char *name, bool default_value = false)

> Retrieves a boolean value from an environment variable.
>
> This function fetches the value of a named environment variable and converts it to a boolean. The conversion is case-insensitive and accepts “true”, “1”, or “on” as true, any other values are considered false. If the environment variable is not set or its value is not recognized as ‘true’, the function returns a default value.
>
> This function uses std::getenv() to access environment variables. The environment variable to look up is specified by the ‘name’ parameter. The value of the environment variable is converted to a lower-case string, and compared to the known ‘true’ strings.
>
> The function does not throw an exception if the environment variable is not found or if the value does not match any of the expected ‘true’ strings.

Parameters

- **name** – The name of the environment variable to look up.
- **default_value** – The value to return if the environment variable is not set or its value does not match any of the known ‘true’ strings. The default is ‘false’.

Returns

true if the environment variable is set and its value is recognized as ‘true’, and false otherwise. If the environment variable is not set, the function returns ‘default_value’.

static int64_t get_int_env_var(const char *name, int64_t default_value = 0)

Retrieves an integer value from an environment variable.

Parameters

- **name** – The name of the environment variable to look up.
- **default_value** – The value to return if the environment variable is not set or its value is not a valid integer. The default is 0.

Returns

The value of the environment variable, converted to an integer. If the environment variable is not set or its value is not a valid integer, the function returns ‘default_value’.

static uint64_t parse_memory_size(const std::string &size_str)

Parses a string representing a memory size and returns its value in bytes.

This method takes a string in the format of “XGi” or “XMi”, where X is a numerical value, and Mi/Gi represents Mebibytes/Gibibytes. The method converts this string into an equivalent memory size in bytes.

The string is case-insensitive, meaning that “1gi” is considered equivalent to “1Gi”.

The string is expected to represent a positive integer followed by either “M” or “G”, signifying mebibytes or gibibytes, respectively. If the string does not follow this format, the behavior is undefined.

Parameters

size_str – A string representing a memory size in mebibytes or gibibytes. For example, “1Gi” represents one gibibyte and “500Mi” represents 500 mebibytes.

Returns

The memory size represented by size_str, converted into bytes. For example, if size_str is “1Gi”, the return value will be 1,073,741,824 (1024 * 1024 * 1024). If size_str is “500Mi”, the return value will be 524,288,000 (500 * 1024 * 1024).

static void set_ucx_to_exclude_cuda_ipc()

Set UCX_TLS to disable cuda_ipc transport

Will check the UCX_TLS environment variable. If it is not already set, it sets UCX_TLS=^cuda_ipc to exclude this transport method. If it is already set and cuda_ipc is not excluded, a warning will be logged.

static void exclude_cuda_ipc_transport_on_igpu()

Disable CUDA interprocess communication (IPC) if the fragment is running on an iGPU.

Tegra devices do not support CUDA IPC.

Calls set_ucx_tls_to_exclude_cuda_ipc if we are running on iGPU.

Friends

friend class service::AppDriverServer

struct DriverMessage

Public Members

DriverMessageCode code

std::any data