Struct CLIOptions

Table of contents

- Defined in File cli_options.hpp

Struct Documentation

struct CLIOptions

CLI Options struct.

This struct is used to store the parsed command line arguments.

Public Functions

void print() const

Print the CLI Options.

Public Members

bool run_driver = false

The flag to run the App Driver.

bool run_worker = false

The flag to run the App Worker.

std::string driver_address

The address of the App Driver.

std::string worker_address

The address of the App Worker.

std::vector<std::string> worker_targets

The list of fragments for the App Worker.

std::string config_path

The path to the configuration file.

Public Static Functions

static std::string resolve_hostname(const std::string &hostname)

Resolve the specified hostname to an IP address, prioritizing IPv4.

This function attempts to resolve the given hostname to an IP address using the system's domain name service (DNS) resolver. It supports both IPv4 and IPv6 addresses and prioritizes IPv4 addresses. If the hostname can be resolved to multiple IP addresses and both IPv4 and IPv6 addresses are available, an IPv4 address will be returned in preference to an IPv6 address.

> **Note**
>
> This function logs an error message using `HOLOSCAN_LOG_ERROR` if `getaddrinfo` fails to resolve the hostname. It also logs the resolved IP address using `HOLOSCAN_LOG_DEBUG` upon successful resolution.

Parameters

hostname – The hostname to be resolved. This is a domain name, such as "example.com".

Returns

A string containing the resolved IP address. If the hostname is resolved to an IPv4 address, the returned value will be in the IPv4 format (e.g., "192.0.2.1"). If no IPv4 address is available and it is an IPv6 address, it will be returned in the IPv6 format (e.g., "2001:db8::1"). If the hostname cannot be resolved or an error occurs, an empty string is returned.

static std::string parse_port(const std::string &address, const std::string &default_port = "")

Extract the port number from the given address. This method supports both IPv4 and IPv6 addresses and can handle addresses with or without square brackets.

The method first checks for the presence of square brackets to determine if the address is an IPv6 address containing a port. If the address does not contain square brackets, it is treated as either an IPv4 address or a hostname, with the port separated by a colon. If multiple colons are present without brackets, the method assumes an IPv6 address without a specified port and will return the default port.

Parameters

- **address** – The address string from which to extract the port. This can be an IPv4 address, an IPv6 address (with or without square brackets), or a hostname.
- **default_port** – The default port to return if the address does not explicitly contain a port number. If this parameter is empty, and no port is found in the address, the method will return an empty string.

Returns

A string representing the extracted port number. If no port can be extracted and a default port is provided, the default port is returned. If no default port is provided and no port can be extracted, an empty string is returned.

static std::pair<std::string, std::string> parse_address(const std::string &address, const std::string &default_ip, const std::string &default_port, bool enclose_ipv6 = false, bool resolve_hostname = true)

Parse the provided address string and extract the IP address and port.

This function supports both IPv4 and IPv6 address formats. When a port is specified, IPv6 addresses are expected to be enclosed in square brackets. If the address string is empty or does not specify an IP address or port, the default values are returned, determined by the deduced IP version from the address format. IPv6 addresses will be enclosed in square brackets if a port is specified and if the `enclose_ipv6` parameter is set to true.

If the IP address is identified as a hostname and the `resolve_hostname` parameter is set to true, the function will resolve the hostname to an IP address.

Parameters

- **address** – The address string to parse, which can contain an IPv4 address, an IPv6 address, or a hostname with an optional port. IPv6 addresses with a port must be enclosed in square brackets (e.g., "[::1]:80"). If the first segment is a colon, it indicates that no IP address is provided before the port, and thus the default IP address is used.

- **default_ip** – The default IP address to return if the address string is empty, does not contain an IP address, or when an IP address cannot be deduced from the address string. This default value is used for both IPv4 and IPv6 addresses when they are not specified in the address string.

- **default_port** – The default port to return if the address string does not specify a port. This should be a string representing a valid port number.

- **enclose_ipv6** – Determines whether to enclose the IPv6 address in square brackets when a port is specified. This parameter is effective only if the address is detected as an IPv6 address and the deduced port is not empty. It defaults to false, meaning that IPv6 addresses will not be enclosed in brackets.

- **resolve_hostname** – Determines whether to resolve the hostname to an IP address. If this parameter is true and the address string contains a hostname, the hostname will be resolved to an IP address. The default value is true.

Returns

A pair containing the extracted IP address and port. The IP address is returned in the first element of the pair, and the port in the second. If the input does not specify an IP address or port, the provided defaults are returned. If the IP address is IPv6, a port is specified, and `enclose_ipv6` is true, the IP address will be enclosed in square brackets.