

# async\_select

## *User Manual*



**MICROEJ<sup>®</sup>**

Reference:	TLT-XXX-MAN-async_select-async_select
Version:	2.0.2
Revision:	XXX

---

## Confidentiality & Intellectual Property

All rights reserved. Information, technical data and tutorials contained in this document are confidential and proprietary under copyright Law of Industrial Smart Software Technology (IS2T S.A.) operating under the brand name MicroEJ®. Without written permission from IS2T S.A., *copying or sending parts of the document or the entire document by any means to third parties is not permitted*. Granted authorizations for using parts of the document or the entire document do not mean IS2T S.A. gives public full access rights.

The information contained herein is not warranted to be error-free. IS2T® and MicroEJ® and all relative logos are trademarks or registered trademarks of IS2T S.A. in France and other Countries.

Java™ is Sun Microsystems' trademark for a technology for developing application software and deploying it in cross-platform, networked environments. When it is used in this documentation without adding the ™ symbol, it includes implementations of the technology by companies other than Sun.

Java™, all Java-based marks and all related logos are trademarks or registered trademarks of Sun Microsystems Inc, in the United States and other Countries.

Other trademarks are proprietary of their authors.

---

---

# Table of Contents

1. Data Structure Documentation .....	1
1.1. async_select_Request struct Reference .....	1
1.1.1. Data Fields .....	1
1.1.2. Field Documentation .....	1
2. File Documentation .....	2
2.1. async_select.h File Reference .....	2
2.1.1. Enumerations .....	2
2.1.2. Functions .....	2
2.2. async_select_configuration.h File Reference .....	3
2.2.1. Macros .....	3
2.3. async_select.c File Reference .....	4
2.3.1. Data Structures .....	4
2.3.2. Macros .....	5
2.3.3. Typedefs .....	5
2.3.4. Variables .....	5
2.3.5. Functions .....	5
2.4. async_select_osal.c File Reference .....	7
2.4.1. Variables .....	7
2.4.2. Functions .....	7

---

# Chapter 1. Data Structure Documentation

## 1.1. `async_select_Request` struct Reference

### 1.1.1. Data Fields

- `int32_t fd`
- `int32_t java_thread_id`
- `int64_t absolute_timeout_ms`
- `SELECT_Operation operation`
- `struct async_select_Request * next`

An asynchronous select request.

### Detailed Description

Sanity check between the expected version of the configuration and the actual version of the configuration. If an error is raised here, it means that a new version of the CCO has been installed and the configuration `async_select_configuration.h` must be updated based on the one provided by the new CCO version.

Definition at line 45 of file `async_select.c`

The Documentation for this struct was generated from the following file:

- `async_select.c`

### 1.1.2. Field Documentation

---

# Chapter 2. File Documentation

## 2.1. async\_select.h File Reference

```
#include <stdint.h>
```

```
#include <sni.h>
```

### 2.1.1. Enumerations

- enum SELECT\_Operation {  
    SELECT\_READ,  
    SELECT\_WRITE  
}

*Select operations list.*

### 2.1.2. Functions

- int32\_t non\_blocking\_select ( int32\_t fd, SELECT\_Operation operation)

*Execute a select() for the given file descriptor and operation without blocking.*

- int32\_t async\_select ( int32\_t fd, SELECT\_Operation operation, int64\_t timeout\_ms, SNI\_callback callback)

*Executes asynchronously a select() operation for the given file descriptor. This function will suspend the execution of the current Java thread using SNI\_suspendCurrentJavaThreadWithCallback(). Once the select() succeeds the Java thread is resumed and the given SNI callback is called.*

- int32\_t async\_select\_init ( void )

*Initialize the async\_select component. This function must be called prior to any call of async\_select().*

- void async\_select\_notify\_closed\_fd ( int32\_t fd)

*Notifies the async\_select task that a file descriptor has been closed. On some systems the close of a file descriptor does not unblock the select that's why we need to notify the async\_select task.*

## Detailed Description

Asynchronous network select API.

Author: . MicroEJ Developer Team

Version: . 2.0.2

Date: . 13 November 2020

Definition in file C:/Jenkins/workspace/M0172\_CCO-Async-Select/bsp-async\_select/target~/ccomponentWorking/bsp/net/inc/async\_select.h

## 2.2. async\_select\_configuration.h File Reference

```
#include <stdint.h>
```

```
#include <sni.h>
```

### 2.2.1. Macros

- #define ASYNC\_SELECT\_CONFIGURATION\_VERSION (2)

*Compatibility sanity check value. This define value is checked in the implementation to validate that the version of this configuration is compatible with the implementation.*

- #define MAX\_NB\_ASYNC\_SELECT (16)

*Maximum number of asynchronous select that can be done at the same moment.*

- #define ASYNC\_SELECT\_TASK\_STACK\_SIZE (2048)

*async\_select task stack size in bytes.*

- #define ASYNC\_SELECT\_TASK\_NAME "AsyncSelect"

*async\_select task name.*

- #define ASYNC\_SELECT\_TASK\_PRIORITY (12)

*async\_select task priority.*

- #define ASYNC\_SELECT\_MUTEX\_NAME "AsyncSelectMutex"

*async\_select mutex name.*

- #define ASYNC\_SELECT\_POLLING\_MODE\_TIMEOUT\_MS (100)

*Timeout in milliseconds used when the async\_select task cannot allocate a socket for notifications.*

- #define ASYNC\_SELECT\_CLOSE\_UNBLOCK\_SELECT

*Set this define if a file descriptor close unblocks the select.*

## Detailed Description

Asynchronous network select configuration.

Author: . MicroEJ Developer Team

Version: . 2.0.2

Date: . 13 November 2020

Definition in file C:/Jenkins/workspace/M0172\_CCO-Async-Select/bsp-async\_select/target~/ccomponentWorking/bsp/net/inc/async\_select\_configuration.h

## 2.3. async\_select.c File Reference

```
#include "async_select.h"
```

```
#include "async_select_configuration.h"
```

```
#include <string.h>
```

```
#include <sys/socket.h>
```

```
#include <sys/select.h>
```

```
#include <netinet/in.h>
```

```
#include <stdbool.h>
```

```
#include <unistd.h>
```

```
#include "LLNET_Common.h"
```

### 2.3.1. Data Structures

- struct async\_select\_Request

*An asynchronous select request.*

## 2.3.2. Macros

- `#define async_select_get_current_time_ms LLMJVM_IMPL_getCurrentTime__Z(1)`

## 2.3.3. Typedefs

- `typedef struct async_select_Request async_select_Request`

*An asynchronous select request.*

## 2.3.4. Variables

- `static async_select_Request all_requests`

*Pool of requests. Used to reserve MAX\_NB\_ASYNC\_SELECT async select requests.*

- `static async_select_Request * free_requests_fifo`

*Linked-list of free requests that can be allocated using `async_select_allocate_request()`.*

- `static async_select_Request * used_requests_fifo`

*Linked-list of used requests.*

- `static fd_set read_fds`

*File descriptor set for `SELECT_READ` requests.*

- `static fd_set write_fds`

*File descriptor set for `SELECT_WRITE` requests.*

- `static volatile int32_t notify_fd_cache`

*Used to unblock `select()` function call.*

- `static volatile uint8_t async_select_fifo_initialized`

*set to one once the FIFOs are initialized.*

## 2.3.5. Functions

- `void async_select_lock ( void )`

*Enter critical section for the `async_select` component.*

- `void async_select_unlock ( void )`

*Exit critical section for the `async_select` component.*

- `int64_t LLMJVM_IMPL_getCurrentTime__Z ( uint8_t system)`



*External function used to retrieve currentTime (defined in LLMJVM)*

- static void async\_select\_do\_select ( void )

*Executes the select() operation for the file descriptors referenced by the received requests.*

- static void async\_select\_update\_notified\_requests ( void )

*After the execution of the select() operation, update the status of the requests that have been notified by the select() or have reached the timeout.*

- static int32\_t async\_select\_get\_notify\_fd ( void )

*Returns the file descriptor created just to unlock the select() when we want to notify the async\_select task that a new request has been sent.*

- static async\_select\_Request \* async\_select\_allocate\_request ( void )

*Find a free request and returns it. The returned request is not put in the used requests FIFO. It must be either put in the used requests FIFO using async\_select\_send\_new\_request() or put back in the free requests FIFO on error using async\_select\_free\_unused\_request().*

- static async\_select\_Request \* async\_select\_free\_used\_request ( async\_select\_Request \* request, async\_select\_Request \* previous\_request\_in\_used\_fifo )

*Remove the given request from the used FIFO and put it in the free FIFO.*

- static void async\_select\_free\_unused\_request ( async\_select\_Request \* request )

*Put the given request in the free FIFO. The request must not be in the used FIFO.*

- static int32\_t async\_select\_send\_new\_request ( async\_select\_Request \* request )

*Notifies the async\_select task that a new request must be managed.*

- static void async\_select\_notify\_select ( void )

*Unlock the select operation.*

- void async\_select\_request\_fifo\_init ( void )

*Initializes the requests FIFOs. This function must be called prior to any call of async\_select(). It can be called several times.*

- int32\_t non\_blocking\_select ( int32\_t fd, SELECT\_Operation operation )

*Execute a select() for the given file descriptor and operation without blocking.*

- int32\_t async\_select ( int32\_t fd, SELECT\_Operation operation, int64\_t timeout\_ms, SNI\_callback callback )

*Executes asynchronously a select() operation for the given file descriptor.*

- void async\_select\_notify\_closed\_fd ( int32\_t fd )

*Notifies the `async_select` task that a file descriptor has been closed. On some systems the close of a file descriptor does not unblock the select that's why we need to notify the `async_select` task.*

- `void async_select_task_main ( )`

*The entry point for the `async_select` task. This function must be called from a dedicated task.*

## Detailed Description

Asynchronous network select implementation.

Author: . MicroEJ Developer Team

Version: . 2.0.2

Date: . 13 November 2020

Definition in file `C:/Jenkins/workspace/M0172_CCO-Async-Select/bsp-async_select/target~/ccomponentWorking/bsp/net/src/async_select.c`

## 2.4. `async_select_osal.c` File Reference

```
#include "async_select.h"
```

```
#include "async_select_configuration.h"
```

```
#include "osal.h"
```

```
#include <stddef.h>
```

### 2.4.1. Variables

- static `OSAL_task_handle_t async_select_task`

*`async_select` OS task.*

- static `OSAL_mutex_handle_t async_select_mutex`

*Mutex used for critical sections.*

### 2.4.2. Functions

- `void async_select_request_fifo_init ( void )`

*Initializes the requests FIFOs. This function must be called prior to any call of `async_select()`. It can be called several times.*

- `void async_select_task_main ( void )`

*The entry point for the `async_select` task. This function must be called from a dedicated task.*

- `void async_select_lock ( void )`

*Enter critical section for the `async_select` component.*

- `void async_select_unlock ( void )`

*Exit critical section for the `async_select` component.*

- `static int32_t async_select_start_task ( void )`

*Start RTOS task and init RTOS specific structures.*

- `OSAL_task_stack_declare ( async_select_task_stack , ASYNC_SELECT_TASK_STACK_SIZE )`

*Stack of the `async_select` task.*

- `int32_t async_select_init ( )`

*Initialize the `async_select` component. This function must be called prior to any call of `async_select()`.*

## Detailed Description

Asynchronous network select implementation over OSAL API.

Author: . MicroEJ Developer Team

Version: . 2.0.2

Date: . 13 November 2020

Definition in file C:/Jenkins/workspace/M0172\_CCO-Async-Select/bsp-async\_select/target~/ccomponentWorking/bsp/net/src/async\_select\_osal.c