



# Application Queue API Reference Manual

JN-RM-2025  
Revision 1.0  
21-Sep-2006



---

## Contents

<b>About this Manual</b>	<b>4</b>
Organisation	4
Conventions	4
Acronyms and Abbreviations	4
Related Documents	4
Feedback Address	4
<b>1 Introduction</b>	<b>5</b>
1.1 Architecture	6
1.2 Purpose	7
<b>2 Function Descriptions</b>	<b>9</b>
u32AppQApiInit	10
psAppQApiReadMlmeInd	11
psAppQApiReadMcpsInd	12
psAppQApiReadHwInd	13
vAppQApiReturnMlmeIndBuffer	14
vAppQApiReturnMcpsIndBuffer	15
vAppQApiReturnHwIndBuffer	16

---

## About this Manual

This manual describes the Jennic Application Queue API (Application Programming Interface) that can be used to handle interrupts in the Jennic JN51xx wireless microcontroller.

---

## Organisation

This document consists of two chapters:

- [Chapter 1](#) gives a brief introduction to the Application Queue API.
- [Chapter 2](#) provides descriptions of the API functions.

---

## Conventions

Files, folders, functions and parameter types are represented in **bold** type.

Function parameters are represented in *italics* type.

Code fragments are represented in the `Courier` typeface.

---

## Acronyms and Abbreviations

API	Application Programming Interface
MAC	Media Access Control (sub-layer)

---

## Related Documents

[R1]	Jennic IEEE 802.15.4 User Guide [JN-UG-3024]
[R2]	Jennic Integrated Peripherals API Reference Manual [JN-RM-2001]
[R3]	Jennic 802.15.4 Stack API Reference Manual [JN-RM-2002]

---

## Feedback Address

If you wish to comment on this manual, or any other Jennic user documentation, please provide your feedback by writing to us (quoting the manual reference number and version) at the following postal address or e-mail address:

Applications  
Jennic Ltd  
Furnival Street  
Sheffield S1 4QT  
United Kingdom

[doc@jennic.com](mailto:doc@jennic.com)

---

# 1 Introduction

This chapter introduces the purpose of the Application Queue API and its position in the IEEE 802.15.4 software architecture.



**Note:** Use of the Application Queue API is completely optional. You can design your applications to operate with or without this API.



**Caution:** *This API cannot be used with the Jennic ZigBee stack.*

## 1.1 Architecture

The Application Queue API provides a queue-based interface between an application and both the IEEE 802.15.4 stack and the hardware drivers (for the Jennic JN51xx wireless microcontroller):

- The API interacts with the IEEE 802.15.4 stack via the Jennic 802.15.4 Stack API (which sits on top of the 802.15.4 stack).
- The API interacts with the Peripheral Hardware Drivers via the Jennic Integrated Peripherals API (which sits on top of the Peripheral Hardware Drivers).

This architecture is illustrated in Figure 1 below.

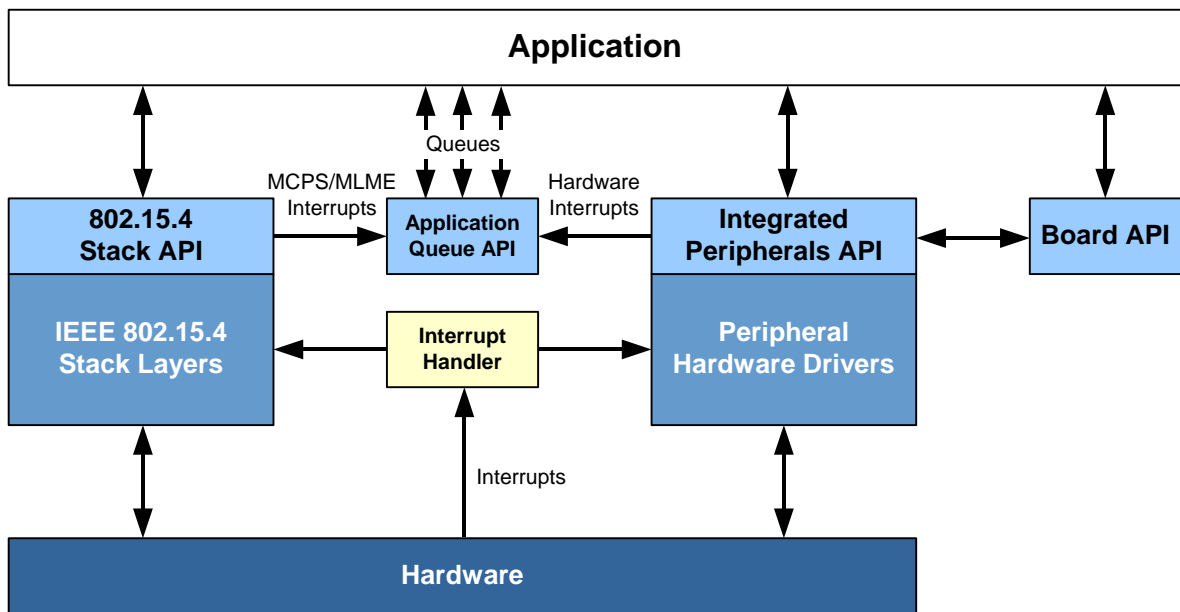


Figure 1: Software Architecture

---

## 1.2 Purpose

The Application Queue API handles interrupts coming from the MAC sub-layer of the IEEE 802.15.4 stack and from the integrated peripherals of the Jennic JN51xx wireless microcontroller, saving the application from dealing with interrupts directly. The API implements a queue for each of three types of interrupt:

- MCPS (MAC Data Services) interrupts coming from the stack
- MLME (MAC Management Services) interrupts coming from the stack
- Hardware interrupts coming from the hardware drivers

The application polls these queues for entries and then processes the entries.



**Note:** The Application Queue API allows callbacks to be defined by the application, as with the normal IEEE 802.15.4 Stack API, but an application can be designed such that they are not necessary.





---

## 2 Function Descriptions

This chapter provides descriptions of the individual functions of the Application Queue API.

The functions are listed below along with their page references.

<b>Function</b>	<b>Page</b>
<a href="#">u32AppQApiInit</a>	10
<a href="#">psAppQApiReadMlmeInd</a>	11
<a href="#">psAppQApiReadMcpsInd</a>	12
<a href="#">psAppQApiReadHwInd</a>	13
<a href="#">vAppQApiReturnMlmeIndBuffer</a>	14
<a href="#">vAppQApiReturnMcpsIndBuffer</a>	15
<a href="#">vAppQApiReturnHwIndBuffer</a>	16

---

**u32AppQApilnit**

---

```
PUBLIC uint32 u32AppQApilnit(  
    PR_QIND_CALLBACK prMlmeCallback,  
    PR_QIND_CALLBACK prMcpsCallback,  
    PR_HWQINT_CALLBACK prHwCallback);
```

### Description

This function initialises the Application Queue API, as well as the underlying 802.15.4 Stack API and hence the whole 802.15.4 stack. The function creates queues for storing a number of upward messages (MLME indications and confirmations, MCPS indications and confirmations, Integrated Peripherals API indications) and registers itself with the lower layers so that all such messages go through it. The function refers to callback functions for the three queues.

The prototype for the MCPS and MLME callbacks is a function that takes no parameters and returns void. The prototype for the hardware indications takes two 32-bit values as parameters and returns void.

### Parameters

<i>prMlmeCallback</i>	Pointer to optional callback function for upward MLME indications and confirmations. If a callback is not required, a value of NULL must be used.
<i>prMcpsCallback</i>	Pointer to optional callback function for upward MCPS indications and confirmations. If a callback is not required, a value of NULL must be used.
<i>prHwCallback</i>	Pointer to optional callback function for upward indications from the Integrated Peripherals API. If a callback is not required, a value of NULL must be used.

### Returns

0 if initialisation failed.

Otherwise, the 32-bit version number of the IEEE 802.15.4 stack (most significant 16 bits are major revision, least significant 16 bits are patch revision/minor revision).

---

## psAppQApiReadMlmeInd

---

```
PUBLIC MAC_MlmeDcfmInd_s *psAppQApiReadMlmeInd(void);
```

### Description

This function enables the application to poll the MLME indication/confirmation queue. If an event is present in the queue, the application can process it. Once processing has finished, the buffer (that contained the event) must be returned to the Application Queue API using the **vAppQApiReturnMlmeIndBuffer()** function. The result is returned in the structure `MAC_MlmeDcfmInd_s`; for details of this structure, refer to the *Jennic 802.15.4 Stack API Reference Manual*.

### Parameters

None

### Returns

`MAC_MlmeDcfmInd_s *`

Pointer to a buffer containing an MLME indication or confirmation, or NULL if the queue is empty.

---

**psAppQApiReadMcpsInd**

---

```
PUBLIC MAC_McpsDcfmInd_s *psAppQApiReadMcpsInd(void);
```

**Description**

This function enables the application to poll the MCPS indication/confirmation queue. If an event is present in the queue, the application can process it. Once processing has finished, the buffer (that contained the event) must be returned to the Application Queue API using the **vAppQApiReturnMcpsIndBuffer()** function. The result is returned in the structure `MAC_McpsDcfmInd_s`; for details of this structure, refer to the *Jennic 802.15.4 Stack API Reference Manual*.

**Parameters**

None

**Returns**

`MAC_McpsDcfmInd_s *`

Pointer to a buffer containing an MCPS indication or confirmation, or NULL if the queue is empty.

---

## psAppQApiReadHwInd

---

```
PUBLIC AppQApiHwInd_s * psAppQApiReadHwInd (void);
```

### Description

This function enables the application to poll the hardware indication queue. If an event is present in the queue, the application can process it. Once processing has finished, the buffer (that contained the event) must be returned to the Application Queue API using the **vAppQApiReturnHwIndBuffer()** function. The result is returned in the structure `AppQApiHwInd_s`, detailed below.

### Parameters

None

### Returns

`AppQApiHwInd_s`, which has the following definition:

```
typedef struct  
{  
    uint32 u32DeviceId;  
    uint32 u32ItemBitmap;  
} AppQApiHwInd_s;
```

`u32DeviceId` and `u32ItemBitmap` are detailed in the *Jennic Integrated Peripherals API Reference Manual*.

---

**vAppQApiReturnMlmeIndBuffer**

---

```
PUBLIC void vAppQApiReturnMlmeIndBuffer(  
    MAC_MlmeDcfmInd_s *psBuffer);
```

**Description**

This function allows the application to return an MLME buffer previously passed up to the application. Once returned, the buffer can be re-used to store and pass another message.

**Parameters**

*\*psBuffer*                      Pointer to MLME buffer to be returned.

**Returns**

None

---

## vAppQApiReturnMcpsIndBuffer

---

```
PUBLIC void vAppQApiReturnMcpsIndBuffer(  
    MAC_McpsDcfmInd_s *psBuffer);
```

### Description

This function allows the application to return an MCPS buffer previously passed up to the application. Once returned, the buffer can be re-used to store and pass another message.

### Parameters

*\*psBuffer*                      Pointer to MCPS buffer to be returned.

### Returns

None

---

**vAppQApiReturnHwIndBuffer**

---

```
PUBLIC void vAppQApiReturnHwIndBuffer(  
    AppQApiHwInd_s *psBuffer);
```

**Description**

This function allows the application to return a hardware event buffer previously passed up to the application from the Integrated Peripherals API. Once returned, the buffer can be re-used to store and pass another message.

**Parameters**

*\*psBuffer*                      Pointer to hardware event buffer to be returned.

**Returns**

None



## Revision History

Version	Date	Description
1.0	21-Sep-2006	First release (the information in this manual was originally contained in the <i>Jennic JN5121-EK000 Demonstration Application Reference Manual, JN-RM-2004</i> ).

## Disclaimer

The contents of this document are subject to change without notice. Customers are advised to consult with JENNIC commercial representatives before ordering.

The information and circuit diagrams in this document are presented as examples of semiconductor device applications, and are not intended for incorporation in devices for actual use. Also, JENNIC is unable to assume responsibility for infringement of any patent rights or other rights of third parties arising from the use of this information or circuit diagrams.

No license is granted by its implication or otherwise under any patent or patent rights of JENNIC Ltd

"Typical" parameters that are provided in this document may vary in different applications and performance may vary over time. All operating parameters must be validated for each customer application by the customer's own technical experts.

**!** *Customers considering the use of our products in special applications where failure or abnormal operation may directly affect human lives or cause physical injury or property damage, or where extremely high levels of reliability are demanded (such as aerospace systems, atomic energy controls, vehicle operating controls, medical devices for life support, etc.) are requested to consult with JENNIC representatives before such use. JENNIC customers using or selling products incorporating JENNIC IP for use in such applications do so at their own risk and agree to fully indemnify JENNIC for any damages resulting from such improper use or sale.*



### Corporate Headquarters

Furnival Street  
Sheffield  
S1 4QT  
United Kingdom

Tel +44 (0)114 281 2655  
Fax +44 (0)114 281 2951  
E-mail [info@jennic.com](mailto:info@jennic.com)

### United States Sales Office

1322 Scott Street, Suite 203  
Point Loma  
CA 92106  
USA

Tel +1 619 223 2215  
Fax +1 619 223 2081  
E-mail [info@us.jennic.com](mailto:info@us.jennic.com)

### Japan Sales Office

Osakaya building 4F  
1-11-8 Higashigotanda  
Shinagawa-ku, Tokyo  
141-0022, Japan

Tel +81 3 5449 7501  
Fax +81 3 5449 0741  
E-mail [info@jp.jennic.com](mailto:info@jp.jennic.com)

### United States Sales Office

1060 First Avenue, Suite 400  
King of Prussia  
PA 19406  
USA

Tel +1 619 223 2215  
Fax +1 619 223 2081  
E-mail [info@us.jennic.com](mailto:info@us.jennic.com)

### Taiwan Sales Office

19F-1, 182, Sec.2  
Tun Hwa S. Road  
Taipei 106  
Taiwan

Tel +886 2 2735 7357  
Fax +886 2 2739 5687  
E-mail [info@tw.jennic.com](mailto:info@tw.jennic.com)