

# Intel® Atom™ Developer Program SDK C Language API Reference

---

November 2009

**Notice:**

Information in this document is provided in connection with Intel® products. No license, express or implied, by estoppels or otherwise, to any intellectual property rights is granted by this document. Except as provided in Intel's Terms and Conditions of Sale for such products, Intel assumes no liability whatsoever, and Intel disclaims any express or implied warranty, relating to sale and/or use of Intel products including liability or warranties relating to fitness for a particular purpose, merchantability, or infringement of any patent, copyright or other intellectual property right. Intel products are not intended for use in medical, life saving, or life sustaining applications.

Intel may make changes to specifications and product descriptions at any time, without notice

The API and software may contain design defects or errors known as errata which may cause the product to deviate from published specifications. Current characterized errata are available on request.

This document and the software described in it are furnished under license and may only be used or copied in accordance with the terms of the license. The information in this document is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Intel Corporation. Intel Corporation assumes no responsibility or liability for any errors or inaccuracies that may appear in this document or any software that may be provided in association with this document. Except as permitted by such license, no part of this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means without the express written consent of Intel Corporation.

Contact your local Intel sales office or your distributor to obtain the latest specifications and before placing your product order.

Copies of documents which have an ordering number and are referenced in this document or other Intel literature may be obtained by calling 1-800-548-4725 or by visiting Intel's website at <http://www.intel.com>.

Copyright © 2009 Intel Corporation.

\* Third party names and brands may be claimed as the property of others.

## Introduction

---

This document is the reference for the Intel® Atom™ Developer Program SDK C Language API. It covers API\_LEVEL 1, API\_VERSION 0.9.

## Framework

---

### SDK API Versioning

There are two constants which provide information about the version of the Application Library. The **ADP\_API\_VERSION** constant identifies the version of the API. The **ADP\_API\_LEVEL** constant identifies the relative version of the API.

#### Syntax

C

```
const wchar_t* ADP_API_VERSION;  
const unsigned long ADP_API_LEVEL;
```

#### Remarks

The **ADP\_API\_VERSION** string is a human readable string of the API version, ie. “1.23.344-beta”. The **ADP\_API\_LEVEL** constant is used to determine compatibility between the Application libraries and the Client Agent. Although the API Level of an application’s Application Library must be the same as the Client Agent, the Client Agent can support a range of **ADP\_API\_LEVELs** concurrently. For example, there could be several releases of the Application libraries, each with a different **ADP\_API\_VERSION** value, that share the same **ADP\_API\_LEVEL**.

If the API version of the Application Library and the Client Agent are incompatible, **ADP\_Initialize()** returns **ADP\_INCOMPATIBLE\_VERSION**. An application and its components must be the same **ADP\_API\_LEVEL**, or else the application will fail to initialize.

### Initialize Function

The **ADP\_Initialize()** function is used to set up internal data structures, and determine whether the system is compatible.

#### Syntax

C

```
ADP_RET_CODE ADP_Initialize ( void );
```

#### Parameters

None

#### Return Value

Value	Meaning
<b>ADP_SUCCESS</b>	Function call was successful.
<b>ADP_NOT_AVAILABLE</b>	The Client Agent is not installed or is not running.
<b>ADP_INCOMPATIBLE_VERSION</b>	The library and Client Agent are incompatible. They don't have to be the same version, just compatible at the call level. See SDK API Versioning.
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

#### Remarks

This function must be the first function called else the other functions will fail with the return code **ADP\_NOT\_INITIALIZED**. If function does not return **ADP\_SUCCESS** the application should exit.

## Close Function

The **ADP\_Close()** function is used to release internal data structures, and shutdown connections used by the library.

#### Syntax

C

```
ADP_RET_CODE ADP_Close ( void );
```

#### Parameters

None

#### Return Value

Value	Meaning
<b>ADP_SUCCESS</b>	Function call was successful.
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, or communication.

#### Remarks

If the function succeeds, the return value is **ADP\_SUCCESS**, else it returns **ADP\_FAILURE** at which point the system is in an indeterminate state.

## ReportCrash Function

The **ADP\_ReportCrash** function is used by an application to report a crash. The Client Agent is responsible for delivering the crash report and the dump information to the Store Backend where the developer can view and assess the information.

#### Syntax

## C

```

typedef struct {
    wchar_t *name;
    wchar_t *value;
} ADP_CrashReportField;

ADP_RET_CODE ADP_ReportCrash(
    const wchar_t      *Module,
    unsigned long      LineNumber,
    const wchar_t      *Message,
    const wchar_t      *Category,
    const wchar_t      *ErrorData,
    unsigned long      ErrorDataSize,
    ADP_CrashReportField *CustomFields,
    unsigned long      CustomFieldNumber
);

```

**Parameters***Module* [in, required]

The name of the application module where the error/crash occurred.

*LineNumber* [in, required]

The line number where the error/crash occurred.

*Message* [in, required]

A short message describing the error/crash or some basic information.

*Category* [in, required]

A text field that can be used for sorting and grouping crash reports in the Developer Dashboard.

*ErrorData* [in, optional]

The detailed error data in text format to be attached to the crash record.

*ErrorDataSize* [in, required if *ErrorData* is provided]

The size of data to be attached to the crash record.

*CustomFields* [in, optional]

The customized data field to be attached to the crash record, in the format of **ADP\_CrashReportField**.

*CustomFieldNumber* [in, required if *CustomFields* is provided].

The number of customized data fields to be attached to the crash record.

**Return Value**

Value	Meaning
-------	---------

<b>ADP_SUCCESS</b>	Function call was successful.
<b>ADP_ERR_DATA_TOO_BIG</b>	<i>ErrorDataSize</i> exceeds 4000 characters.
<b>ADP_NOT_AVAILABLE</b>	Returned if the Client Agent “goes away” while the application is running. The application call <b>ADP_Initialize()</b> before trying again.
<b>ADP_NOT_INITIALIZED</b>	Returned if the <b>ADP_Initialize()</b> function has not been called.
<b>ADP_NOT_AUTHORIZED</b>	Returned if the <b>ADP_IsAppAuthorized()</b> function has not been called with return code <b>ADP_AUTHORIZED</b> .
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

### Remarks

*ErrorDataSize* should not exceed 4000 characters.

*CustomFieldNumber* should not exceed 20.

*Module*, *Message*, and *Category* as well as *name* and *value* fields in *CustomFields* should not exceed 80 characters.

Other part of the system (Client Agent, etc.,) will add additional information to the Crash Record prior to sending to the Back End. For example: timestamps, OS, etc. It is the developer’s responsibility to catch fatal errors or crashes and call the **ADP\_ReportCrash()** function.

**NOTE: Components should not call ADP\_ReportCrash(), only Applications.**

## Applications

---

### IsAuthorized Function

The **ADP\_IsAuthorized** function determines whether the application is authorized to run. It should be called by an application after a successful call to **ADP\_Initialize()**. Calling any Application Library function (besides **ADP\_Initialize()**) without a successful call to **ADP\_IsAuthorized ()** (return code = **ADP\_AUTHORIZED**) will result in an error: **ADP\_NOT\_AUTHORIZED**.

### Syntax

C

```
ADP_RET_CODE ADP_IsAuthorized(
    ADP_APPLICATIONID ApplicationId
);
```

## Parameters

*ApplicationId* [in, required]

The unique id generated by the Developer Portal when a developer creates the application in the site. The developer copies the value from the site into this argument.

## Return Value

Value	Meaning
<b>ADP_AUTHORIZED</b>	Application is authorized to run.
<b>ADP_NOT_AUTHORIZED</b>	Application is not authorized to run.
<b>ADP_AUTHORIZATION_EXPIRED</b>	Application's license has not been refreshed in the required time. It still may be valid.
<b>ADP_NOT_AVAILABLE</b>	Returned if the Client Agent "goes away" while the application is running. The application needs to call <b>ADP_Initialize()</b> before trying again.
<b>ADP_NOT_INITIALIZED</b>	Returned if the <b>ADP_Initialize()</b> function has not been called.
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

## Remarks

It is not an error to call **ADP\_IsAuthorized()** more than once, but only the first Application ID is used. The Application ID passed in is used by subsequent Application Library functions, ie. **ADP\_ReportCrash()**.

## Components

---

### IsAppAuthorized Function

The **ADP\_IsAppAuthorized** function determines whether an application is authorized to use a component. It is the responsibility of the component developer to call the function and communicate the results to the application.

#### Syntax

C

```
ADP_RET_CODE ADP_IsAppAuthorized(
    ADP_COMPONENTID ComponentId
);
```

#### Parameters

*ComponentId* [in]

The ID of the component.

#### Return Value

Value	Meaning
<b>ADP_AUTHORIZED</b>	Application is authorized to use the component.
<b>ADP_NOT_AUTHORIZED</b>	Application is not authorized to use the component.
<b>ADP_AUTHORIZATION_EXPIRED</b>	Application's license to use the component has not been refreshed in the required time. It still may be valid.
<b>ADP_NOT_AVAILABLE</b>	Returned if the Client Agent "goes away" while the application is running. The application needs to call <b>ADP_Initialize()</b> before trying again.
<b>ADP_NOT_INITIALIZED</b>	Returned if the <b>ADP_Initialize()</b> function has not been called
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

#### Remarks

It is the responsibility of the Component developer to call **ADP\_IsAppAuthorized()**. The component should check for authorized when it was loaded by an application, during its constructor or as part of an initialization sequence. Likewise, it is the responsibility of component developer to handle a situation when call to **ADP\_IsAppAuthorized()** fails, indicating that application is not authorized to use the component.

## Instrumentation

Instrumentation functions allow applications to record "execution" or usage records. Instrumentation can be used by applications to track usage, statistics, etc. It is the Client Agent (based on user "opt-in" preferences) that actually determines whether or not the data collected is actually stored and transmitted to the back-end server.

### ApplicationBeginEvent Function

The **ADP\_ApplicationBeginEvent** function creates an application start usage record.

#### Syntax

C

```
ADP_RET_CODE ADP_ApplicationBeginEvent( void );
```

#### Parameters

None

#### Return Value

Value	Meaning
<b>ADP_SUCCESS</b>	Function call was successful.
<b>ADP_NOT_AVAILABLE</b>	Returned if the Client Agent “goes away” while the application is running. The application needs to call <b>ADP_Initialize()</b> before trying again.
<b>ADP_NOT_INITIALIZED</b>	Returned if the <b>ADP_Initialize()</b> function has not been called.
<b>ADP_NOT_AUTHORIZED</b>	Returned if the <b>ADP_IsAppAuthorized()</b> function has not be called with return code <b>ADP_AUTHORIZED</b> .
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

#### Remarks

A call to this function records that the application was launched. A subsequent call to **ADP\_ApplicationEndEvent()** must be made to record the duration of use.

## ApplicationEndEvent

The **ADP\_ApplicationEndEvent** function creates an application end usage record.

#### Syntax

C

```
ADP_RET_CODE ADP_ApplicationEndEvent( void );
```

#### Parameters

None

#### Return Value

Value	Meaning
<b>ADP_SUCCESS</b>	Function call was successful.
<b>ADP_NO_APP_BEGIN_EVENT</b>	Returned if there has been no call to <b>ADP_ApplicationBeginEvent()</b> .

<b>ADP_NOT_AVAILABLE</b>	Returned if the Client Agent “goes away” while the application is running. The application needs to call <b>ADP_Initialize()</b> before trying again.
<b>ADP_NOT_INITIALIZED</b>	Returned if the <b>ADP_Initialize()</b> function has not be called.
<b>ADP_NOT_AUTHORIZED</b>	Returned if the <b>ADP_IsAppAuthorized()</b> function has not be called with return code <b>ADP_AUTHORIZED</b> .
<b>ADP_FAILURE</b>	Other failure e.g., memory, file, communication.

### Remarks

It is not an error to call this function multiple times, but only the last call is used. A prior call to **ADP\_ApplicationBeginEvent()** is required. This function does not need to be called if the developer just wants to track the number of times the application is used and not how long is it used.