

Licence API

User guide

MAN1016 Revision 4.0.1



PROPRIETARY INFORMATION

The information contained in this document is the property of Aculab plc and may be the subject of patents pending or granted, and must not be copied or disclosed without prior written permission. It should not be used for commercial purposes without prior agreement in writing.

All trademarks recognised and acknowledged.

Aculab plc endeavours to ensure that the information in this document is correct and fairly stated but does not accept liability for any error or omission.

The development of Aculab's products and services is continuous and published information may not be up to date. It is important to check the current position with Aculab plc.

Copyright © Aculab plc. 2004-2017 all rights reserved.

Document Revision

Rev	Date	Detail
3.1.2	13.09.10	First issue
3.2.0	03.02.11	Update for shared object API
3.3.0	10.01.14	Update for IPv6 support
4.0.0	18.01.17	Update for server related operations
4.0.1	06.02.17	Addition of Server Delete Re-Try

1	Introduction	3
2	Connecting to the target product	4
3	Obtaining licence information of the target product	6
4	Obtaining licence information of the features	8
5	Obtaining supported features of the product	10
6	Machine ID generation for the product	11
7	Machine ID generation for a feature	12
8	Licence key installation.....	13
9	Golden key installation.....	14
10	Checking connection with the server.....	15
11	Installing a licence	16
12	Renewing licences	17
13	Finding out how many licences can be renewed	18
14	Getting the installed Licence Keys	19
15	Deleting Licence Keys.....	20
16	Getting licence manager version	22
17	Clearing up the resources	23
	Appendix A: Error codes	24
	A.1 Licence API	25
	A.2 Licence manager.....	26
	Appendix B: Additional functions	27

1 Introduction

Licence API consists of two components `al_details.h` and `LicenceManagerAdminInterface.so` or `LicenceManagerAdminInterface.dll` (and the associated `LicenceManagerAdminInterface.lib` for Windows). The library functions provided facilitate integrating all the functionality provided in LicenceAdmin tool with your own applications. This functionality is described in the following sections. All the API function calls return zero if successful.

2 Connecting to the target product

Call one of the following to initialise the connection:

Function prototype:

```
int al_init( AL_LICENCE_PRODUCT_DETAILS* pLicenceProductDetails );
```

CAUTION

Mandatory function call – supports IPv4 addresses only

Pointer to an AL_LICENCE_PRODUCT_DETAILS data structure is the input to the function. This should be allocated and filled in by the application program.

The members are:

```
char IPAddress[MAX_IP_ADDRESS_LENGTH]; // Copy the IP address of the
target product
unsigned int Port; // Copy the port of the target product
```

Code example:

```
int ReturnCode;
AL_LICENCE_PRODUCT_DETAILS LicenceProductDetails;

memset( &LicenceProductDetails, 0, sizeof( LicenceProductDetails ) );
strncpy( LicenceProductDetails.IPAddress, "127.0.0.1",
MAX_IP_ADDRESS_LENGTH ); // for local host
LicenceProductDetails.Port = 2546;

ReturnCode = al_init( &LicenceProductDetails );
if(ReturnCode)
{
    return ReturnCode; // Error
}
```

Function prototype:

```
int al_init_host( const char * host, unsigned int port );
```

CAUTION

Mandatory function call – supports IPv4 and IPv6 addresses

The input parameters are:

```
const char * host; // Pointer to a zero-terminated string containing the  
IP address of the target product  
unsigned int port; // The IP port number of the target product
```

Code example:

```
int ReturnCode;  
char host = "fd9f:af1c:a25b:0:c9c7:545:a4d:f8cf";  
unsigned int portnum = 2546;  
  
ReturnCode = al_init_host( host, portnum );  
if(ReturnCode)  
{  
    return ReturnCode; // Error  
}
```

3 Obtaining licence information of the target product

Function prototype:

```
int al_get_details( AL_LICENCE_INFORMATION* pLicenceInfo );
```

Pointer to an AL_LICENCE_INFORMATION data structure is the input to the function. This should be allocated by the application program. All the members return parameters describing the licence information of the target product.

The members are:

```
char ProductName[MAX_PRODUCT_NAME_LENGTH]; // Returns the name of the
product

unsigned int NumberOfProductLicences; // Returns the number of product
licence keys installed

unsigned int NumberOfSupportedFeatures; // Returns the number of
supported features of the product

unsigned int TotalProductLicencesAvailable; // Returns the total product
licence units available

unsigned int TotalUsedProductLicence; // Returns the total product licence
units used*
```

NOTE

***This value has a multiplication factor based on the target product. e.g. each licence unit allows the use of two voice channels. Please refer to the documentation of the target product to check the multiplication factor.**

```
unsigned int AdminPrivileges; // Returns if the product has an admin key
installed

unsigned int SupportedFeatures[MAX_SUPPORTED_FEATURES]; // Returns the
supported feature IDs

AL_PRODUCT_LICENCE_DETAILS ProductLicences[LICENCE_ARRAY_SIZE]; // Returns
the details of each product licence key installed. The array will have
NumberOfProductLicences valid entries
```

The members of AL_PRODUCT_LICENCE_DETAILS data structure are,

```
unsigned int SecondsLeft; // Returns the time left in seconds, if this
licence key expires

unsigned int Licences; // Returns the total product licence units
available with this licence key, if limited

unsigned int Expires; // Returns if the product licences available with
this licence key expires or not, non-zero value indicates expiry

unsigned int UnlimitedLicences; // Returns if the number of product
licences installed with this licence key is limited or not, non-zero value
indicates no limit

char Units[UNITS_STR_MAX_LENGTH]; // Returns the name of the product
licence units
```

Code example:

```
int ReturnCode;
AL_LICENCE_INFORMATION LicenseInfo;

memset( &LicenseInfo, 0, sizeof( LicenseInfo ) );

ReturnCode = al_get_details( &LicenseInfo );
if(ReturnCode)
{
    return ReturnCode; // Error
}
else
{
    // Print the product licence information
}
```

4 Obtaining licence information of the features

Function prototype:

```
int al_get_feat_details( AL_FEATURE_INFORMATION* pFeatureInfo );
```

Pointer to an AL_FEATURE_INFORMATION data structure is the input to the function. This should be allocated by the application program. This function may be called LicenseInfo.NumberOfSupportedFeatures number of times, using FeatureNumber member to input the required feature ID. Feature IDs are returned in LicenseInfo.SupportedFeatures array. The remaining members return parameters describing the licence information of the target product feature.

The members are:

```
unsigned int FeatureNumber; // Copy the required feature ID
unsigned int NumberOfFeatureLicences; // Returns the number of feature
licence keys installed
unsigned int TotalFeatureLicencesAvailable; // Returns the total number of
feature licence units available
unsigned int TotalUsedFeatureLicence; // Returns the total feature licence
units used**
```

NOTE

****This value has a multiplication factor based on the target product feature. e.g. each feature licence unit allows the use of two video channels. Please refer to the documentation of the target product to check the multiplication factor.**

```
AL_FEATURE_LICENCE_DETAILS FeatureLicences[LICENCE_ARRAY_SIZE]; // Returns
the details of each feature licence key installed. The array will have
NumberOfFeatureLicences valid entries
```

The members of AL_FEATURE_LICENCE_DETAILS data structure are,

```
unsigned int SecondsLeft; // Returns the time left in seconds, if this
licence key expires
unsigned int Licences; // Returns the total feature licence units
available with this licence key, if limited
unsigned int Expires; // Returns if the feature licences available with
this licence key expires or not, non-zero value indicates expiry
unsigned int UnlimitedLicences; // Returns if the number of feature
licences installed with this licence key is limited or not, non-zero value
indicates no limit
char Units[UNITS_STR_MAX_LENGTH]; // Returns the name of the feature
licence units
```


Code example:

```
int ReturnCode;
AL_FEATURE_INFORMATION FeatureInfo;

for(i=0; i<LicenseInfo.NumberOfSupportedFeatures; i++)
{
    memset( &FeatureInfo, 0, sizeof( FeatureInfo ) );
    FeatureInfo.FeatureNumber = LicenseInfo.SupportedFeatures[i]

    ReturnCode = al_get_feat_details( &FeatureInfo );
    if(ReturnCode)
    {
        return ReturnCode; // Error
    }
    else
    {
        // Print the feature licence information
    }
}
```

5 Obtaining supported features of the product

Function prototype:

```
int al_get_name_units( AL_FEATURE_NAME_UNITS* pFeatureNameUnits );
```

Pointer to an AL_FEATURE_NAME_UNITS data structure is the input to the function. This should be allocated by the application program. This function may be called LicenseInfo.NumberOfSupportedFeatures number of times, using FeatureNumber member to input the required feature ID. Feature IDs are returned in LicenseInfo.SupportedFeatures array. The remaining members return parameters describing the target product feature.

The members are:

```
unsigned int FeatureNumber; // Copy the required feature ID
char Name[MAX_FEATURE_NAME]; // Returns the name of the feature
char Description[MAX_FEATURE_DESCRIPTION]; // Returns a short description
of the feature
char Units[UNITS_STR_MAX_LENGTH]; // Returns the name of the feature
licence units
```

Code example:

```
int ReturnCode;
AL_FEATURE_NAME_UNITS FeatureNameUnits;

for(i=0; i<LicenseInfo.NumberOfSupportedFeatures; i++)
{
    memset( &FeatureNameUnits, 0, sizeof( FeatureNameUnits ) );
    FeatureNameUnits.FeatureNumber = LicenseInfo.SupportedFeatures[i]

    ReturnCode = al_get_name_units( &FeatureNameUnits );
    if(ReturnCode)
    {
        return ReturnCode; // Error
    }
    else
    {
        // Print the feature information
    }
}
```

6 Machine ID generation for the product

Function prototype:

```
int al_prep_for_new_key( AL_MACHINE_ID* mId );
```

Pointer to an AL_MACHINE_ID data structure is the input to the function. This should be allocated by the application program.

The members are:

```
char mId[256]; // Returns the machine ID to obtain a product licence key
```

Code example:

```
int ReturnCode;
AL_MACHINE_ID MachineId;

memset( &MachineId, 0, sizeof( MachineId ) );

ReturnCode = al_prep_for_new_key( &MachineId );
if(ReturnCode)
{
    return ReturnCode; // Error
}
else
{
    // Print the machine ID
}
```

7 Machine ID generation for a feature

Function prototype:

```
int al_prep_for_new_feature( AL_MACHINE_ID* mId );
```

Pointer to an `AL_MACHINE_ID` data structure is the input to the function. This should be allocated by the application program.

The members are:

```
char mId[256]; // Returns the machine ID to obtain a feature licence key
```

Code example:

```
int ReturnCode;
AL_MACHINE_ID MachineId;

memset( &MachineId, 0, sizeof( MachineId ) );

ReturnCode = al_prep_for_new_feature( &MachineId );
if(ReturnCode)
{
    return ReturnCode; // Error
}
else
{
    // Print the machine ID
}
```

8 Licence key installation

Function prototype:

```
int al_install_new_key( AL_LICENCE_KEY* lk );
```

Pointer to an `AL_LICENCE_KEY` data structure is the input to the function. This should be allocated and filled in by the application program. Use to install both product and feature licence keys.

The members are:

```
char alk[256]; // Copy the licence key provided by Aculab
```

Code example:

```
int ReturnCode;
AL_LICENCE_KEY NewKey;

memset( &NewKey, 0, sizeof( NewKey ) );
ReturnCode = al_install_new_key( &NewKey );
if(ReturnCode)
{
    return ReturnCode; // Error
}
```

9 Golden key installation

Function prototype:

```
int al_admin( const char* ak );
```

Copy and pass in the Golden key provided by Aculab as a C string.

Code example:

```
int ReturnCode;  
const char* GoldenKey = "YourGoldenKey";  
  
ReturnCode = al_admin( GoldenKey );  
if(ReturnCode)  
{  
    return ReturnCode; // Error  
}
```

10 Checking connection with the server

Function prototype:

```
int al_Check_server ( void );
```

Check whether the licence manager can communicate with the Aculab licence servers.

Code example:

```
int ReturnCode;  
  
ReturnCode = al_Check_server();  
if(ReturnCode)  
{  
    return ReturnCode; // Error  
}
```

11 Installing a licence

Function prototype:

```
int al_install_new_licence( char *licence, AL_FAIL_STRING *failString);
```

Install a licence. If the Licence Manager cannot communicate with the servers then a string will be returned that can be used with the Aculab Licence Tool (ALT).

Code example:

```
int ReturnCode;
char *licence = "your licence here";
AL_FAIL_STRING failString;

ReturnCode = al_install_new_licence( licence, &failString );

if(ReturnCode)
{
    printf("Fail String Returnend - %s",failString.fs);
    return ReturnCode; // Error
}
```


12 Renewing licences

Function prototype:

```
int al_renew_licences( AL_FAIL_STRING *failString );
```

If licences have been installed that require to be renewed this API will invoke the routine to attempt renewal. If the Licence Manager cannot communicate with the server then a string will be returned to be used with the Aculab Licence Tool (ALT).

Code example:

```
int ReturnCode;
AL_FAIL_STRING failString;

ReturnCode = al_renew_licence( &failString );

if(ReturnCode)
{
    printf("Fail String Returnend - %s", failString.fs);
    return ReturnCode; // Error
}
```

13 Finding out how many licences can be renewed

Function prototype:

```
int al_renew_licences_quantity( AL_QUANTITY_STRING* quant );
```

If licences have been installed that require to be renewed this API will invoke the routine to attempt renewal. If the Licence Manager cannot communicate with the server then a string will be returned to be used with the Aculab Licence Tool (ALT).

Code example:

```
AL_QUANTITY_STRING quantityStruct;  
int Result = al_renew_licences_quantity( &quantityStruct );  
  
if ( Result )  
{  
    printf("\n failed: %d\n", Result);  
}  
else  
    printf("\n    %s Key's can be renewed \n",quantityStruct.quantity);
```

14 Getting the installed Licence Keys

Function prototype:

```
int al_get_licence_key( AL_LICENCE_KEY* lk , unsigned int mode );
```

This function will create a list of licence keys and return them. Make the initial call with mode 0 and subsequent calls with mode 1. An empty string will be returned when all licence keys have been returned.

Code example:

```
AL_LICENCE_KEY NewKey;
int Result = al_get_licence_key( &NewKey , 0 );
if (NewKey.alk[0] == 0)
{
    printf ("\n No licence keys to return");
    return Licences;
}
printf("%s",NewKey.alk);
while (1)
{
    Result = al_get_licence_key( &NewKey , 1 );
    if (NewKey.alk[0] == 0)
        break;
    printf("%s",NewKey.alk);
}
```

15 Deleting Licence Keys

Function prototype:

```
int al_delete_licence_key( AL_LICENCE_KEY* lk , AL_DELETE_STRING* ds );
```

This function can be used to delete licence keys. If the Licence Manager can communicate with the licence servers and the delete is successful then the delete will return the licence that can now be installed elsewhere. If the Licence Manager cannot communicate with the servers then a string that can be used with the Aculab Licence Tool (ALT) will be returned.

Code example:

```
AL_LICENCE_KEY lk;
AL_DELETE_STRING ds;
int result;
strcpy(lk.alk, "your licence key");
result = al_delete_licence_key( &lk , &ds );
if (result)
    printf("there was a problem %d , %s",result, lk.alk );
else
{
    printf(" delete return - %s", ds.dS );
}
```

16 Deleting Licence Key on Server

Function prototype:

```
int al_delete_licence_key_on_server(AL_DELETE_STRING* ds );
```

This function can be used if there was a problem deleting a licence on the servers. If the delete string returned by a licence delete has the sub-string "deleted" in it then the licence was not deleted on the server. If this is a new style permanent licence then it may be that the problem was with the network connection. To attempt to try to delete on the server again use this API call. If the Licence Manager cannot communicate with the servers then a string that can be used with the Aculab Licence Tool (ALT) will be returned.

Code example:

```
AL_LICENCE_KEY lk;
AL_DELETE_STRING ds;
int result;
strcpy(lk.alk, licence.c_str());
std::string log_out = "deleting - " + licence;
result = al_delete_licence_key( &lk , &ds );
if (result)
    printf("there was a problem %d , %s",result, lk.alk );
else
{
    printf(" - %s", ds.dS );
    if (strstr(ds.dS,"deleted")!=NULL)
    {
        printf("\n\nThe licence was deleted but the licence could not be deleted
            on the server. This may be");
        printf("That the sesrver could not be reached or the licence cannot be
            moved. Do you want to ");
        printf("try to delete at the server again ? ");

        if (askDeleteOnServer())
        {
            result = al_delete_licence_key_on_server(&ds);
            if (result)
                printf("there was a problem %d , %s",result, lk.alk );
            else
            {
                printf(" - %s", ds.dS );
            }
        }
    }
}
```

17 Getting licence manager version

Function prototype:

```
int al_get_lm_version( AL_VERSION_STRING* vs );
```

This function can be used to find the version of the licence manager.

Code example:

```
AL_VERSION_STRING vs;  
int result;  
result = al_get_lm_version(&vs);  
if (result)  
    printf("there was a problem");  
else  
    printf("Licence Manager Version - %s", vs.vs );
```

18 Clearing up the resources

Function prototype:

```
int al_stop( void );
```

CAUTION

Mandatory function call

Call once done with the Licence API. `al_init()` or `al_init_host()` may be called multiple times in sequence to connect to different products without calling `al_stop()`.

Appendix A: Error codes

A.1 Licence API

```
#define AL_ERR_WINSOCK_VER           -5000 // Could not find version 2.2 of
WinSock DLL

#define AL_ERR_WINSOCK_USE           -5001 // Could not find a usable WinSock
DLL

#define AL_ERR_SOCKET                -5002 // Failed to create a socket

#define AL_ERR_CONNECT               -5003 // Socket connect failed

#define AL_ERR_COMMS                 -5004 // Communication with the product
failed

#define AL_ERR_LICENCEMANAGER        -5005 // Licence manager returned an
error

#define AL_ERR_BAD_PARAMETER         -5006 // Invalid input parameter(s)

#define AL_ERR_NOT_INITIALISED       -5007 // al_init() is not called or
failed
```

A.2 Licence manager

```
#define AL_GEN_ERROR -6100 // An undefined error
#define AL_ERR_FILE_OPEN -6101 // A file failed to open
#define AL_ERR_FILE_CLOSE -6102 // A file failed to close
#define AL_ERR_FILE_REM -6103 // A file failed to be removed
#define AL_ERR_FILE_RNM -6104 // A file failed to be renamed
#define AL_ERR_SOCKET_INIT -6105 // A socket initialisation failed
#define AL_ERR_SOCKET_SETUP -6106 // A socket setup failed
#define AL_ERR_THREAD -6107 // A thread did not start
#define AL_ERR_KNOWN_KEY -6108 // Licence key has been installed
before
#define AL_ERR_TAMPER -6109 // The licence key has been
tampered with
#define AL_ERR_NO_MANAGER -6110 // No licence manager could be
found
#define AL_ERR_INVALID_PARM -6111 // An invalid parameter has been
given
#define AL_ERR_LICENCE_EXP -6112 // The licence key has expired
#define AL_ERR_LICENCE_AVL -6113 // No licences are available
#define AL_ERR_FILE_EXISTS -6114 // A file to be created already
exists
#define AL_ERR_SOCKET_COMMS -6115 // Communication over a socket has
reported an error
#define AL_ERR_ADMIN_SETUP -6116 // Aculab admin failed
#define AL_ERR_ADMIN_DISALLOWED -6117 // Aculab admin not allowed
#define AL_ERR_ADMIN_INIT -6118 // Aculab admin init failed
#define AL_ERR_KNOWN_FEATURE -6119 // Feature key has been installed
before
#define AL_ERR_NO_LICENSE_IN_USES -6120 // No licenses in uses
#define AL_ERR_KEY_VALIDATION -6121 // Encryption in the Key or Machine
Id doesn't match the expected
#define AL_ERR_ADMIN_KEY_VALIDATION -6122 // Encryption validation failed on
the admin key
#define AL_ERR_INVALID_PORT -6123 // TCP Port is either 0 or greater
than 65535 (0xffff)
#define AL_ERR_VIRTUAL_MACHINE -6124 // Virtual Machine detected. This
is not supported
#define AL_ERR_BAD_VERSION -6125 // Library version incorrect
#define AL_ERR_BAD_LIBRARY -6126 // Not an aculab license manager
```

Appendix B: Additional functions

Function prototype:

```
AL_TIME al_seconds_to_time( unsigned int Seconds );
```

Converts input in seconds to AL_TIME structure and returns a copy.

The members are:

```
unsigned int Days;  
unsigned int Hours;  
unsigned int Minutes;  
unsigned int Seconds;
```

Code example:

```
#include <stdio.h>  
  
int ReturnCode;  
AL_LICENCE_INFORMATION LicenseInfo;  
unsigned int NextLicense = 0;  
AL_TIME TimeStruct;  
  
memset( &LicenseInfo, 0, sizeof( LicenseInfo ) );  
  
ReturnCode = al_get_details( &LicenseInfo );  
if(ReturnCode)  
{  
    return ReturnCode; // Error  
}  
for ( ; NextLicense < LicenseInfo.NumberOfProductLicences; NextLicense++ )  
{  
    TimeStruct = al_seconds_to_time (   
        LicenseInfo.ProductLicences[NextLicense].SecondsLeft  
    );  
  
    printf(  
        "Remaining Time - %d Days %d Hours %d Minutes %d Seconds\n",  
        TimeStruct.Days, TimeStruct.Hours,  
        TimeStruct.Minutes, TimeStruct.Seconds  
    );  
}
```