```
// Common U2F raw message format header - Review Draft
// 2014-10-08
// Editor: Jakob Ehrensvard, Yubico, jakob@yubico.com

#ifndef __U2F_H_INCLUDED__
#define __U2F_H_INCLUDED__

#ifdef _MSC_VER  // Windows
typedef unsigned char     uint8_t;
typedef unsigned short    uint16_t;
typedef unsigned int      uint32_t;
typedef unsigned long int uint64_t;
#else
#include <stdint.h>
#endif

#ifdef __cplusplus
extern "C" {
#endif

// General constants

#define U2F_EC_KEY_SIZE         32      // EC key size in bytes
#define U2F_EC_POINT_SIZE       ((U2F_EC_KEY_SIZE * 2) + 1) // Size of EC point
#define U2F_MAX_KH_SIZE         128     // Max size of key handle
#define U2F_MAX_ATT_CERT_SIZE   2048    // Max size of attestation certificate
#define U2F_MAX_EC_SIG_SIZE     72      // Max size of DER coded EC signature
#define U2F_CTR_SIZE            4       // Size of counter field
#define U2F_APPID_SIZE          32      // Size of application id
#define U2F_CHAL_SIZE           32      // Size of challenge

#define ENC_SIZE(x)             ((x + 7) & 0xfff8)

// EC (uncompressed) point

#define U2F_POINT_UNCOMPRESSED  0x04    // Uncompressed point format

typedef struct {
    uint8_t pointFormat;                // Point type
    uint8_t x[U2F_EC_KEY_SIZE];         // X-value
    uint8_t y[U2F_EC_KEY_SIZE];         // Y-value
} U2F_EC_POINT;

// U2F native commands

#define U2F_REGISTER            0x01    // Registration command
#define U2F_AUTHENTICATE        0x02    // Authenticate/sign command
#define U2F_VERSION             0x03    // Read version string command

#define U2F_VENDOR_FIRST        0xc0    // First vendor defined command
#define U2F_VENDOR_LAST         0xff    // Last vendor defined command

// U2F_CMD_REGISTER command defines

#define U2F_REGISTER_ID         0x05    // Version 2 registration identifier
#define U2F_REGISTER_HASH_ID    0x00    // Version 2 hash identintifier

typedef struct {
    uint8_t chal[U2F_CHAL_SIZE];        // Challenge
    uint8_t appId[U2F_APPID_SIZE];      // Application id
```

```
} U2F_REGISTER_REQ;

typedef struct {
    uint8_t registerId;                 // Registration identifier (U2F_REGISTER_ID_V2)
    U2F_EC_POINT pubKey;                // Generated public key
    uint8_t keyHandleLen;               // Length of key handle
    uint8_t keyHandleCertSig[
        U2F_MAX_KH_SIZE +               // Key handle
        U2F_MAX_ATT_CERT_SIZE +         // Attestation certificate
        U2F_MAX_EC_SIG_SIZE];           // Registration signature
} U2F_REGISTER_RESP;

// U2F_CMD_AUTHENTICATE command defines

// Authentication control byte

#define U2F_AUTH_ENFORCE        0x03    // Enforce user presence and sign
#define U2F_AUTH_CHECK_ONLY     0x07    // Check only
#define U2F_AUTH_FLAG_TUP       0x01    // Test of user presence set

typedef struct {
    uint8_t chal[U2F_CHAL_SIZE];        // Challenge
    uint8_t appId[U2F_APPID_SIZE];      // Application id
    uint8_t keyHandleLen;               // Length of key handle
    uint8_t keyHandle[U2F_MAX_KH_SIZE]; // Key handle
} U2F_AUTHENTICATE_REQ;

typedef struct {
    uint8_t flags;                      // U2F_AUTH_FLAG_ values
    uint8_t ctr[U2F_CTR_SIZE];          // Counter field (big-endian)
    uint8_t sig[U2F_MAX_EC_SIG_SIZE];   // Signature
} U2F_AUTHENTICATE_RESP;

// Command status responses

#define U2F_SW_NO_ERROR         0x9000  // SW_NO_ERROR
#define U2F_SW_WRONG_DATA       0x6984  // SW_WRONG_DATA
#define U2F_SW_CONDITIONS_NOT_SATISFIED 0x6985 // SW_CONDITIONS_NOT_SATISFIED
#define U2F_SW_INS_NOT_SUPPORTED 0x6d00 // SW_INS_NOT_SUPPORTED

#ifdef __cplusplus
}
#endif

#endif  // __U2F_H_INCLUDED__
```