

# **RTCU Deployment Server API**

**Version 3.10**

## Table of Content

|  |    |
|--|----|
| Table of Content .....                     | 2  |
| Introduction .....                         | 4  |
| Graphical illustration of the system ..... | 4  |
| Contents of package .....                  | 5  |
| Functions in the RDSAPI.DLL library .....  | 6  |
| Server Functions .....                     | 6  |
| rdsConnect() .....                         | 6  |
| rdsDisconnect() .....                      | 6  |
| rdsGetStatus() .....                       | 7  |
| rdsGetVersion() .....                      | 7  |
| rdsGetServerVersion() .....                | 7  |
| Callback cbStatus .....                    | 8  |
| rdsGetServerStatus() .....                 | 8  |
| Log functions .....                        | 9  |
| Callback notify_log_data .....             | 9  |
| Callback notify_log_clear .....            | 9  |
| rdsRegisterLogNotification() .....         | 9  |
| rdsLogLoad() .....                         | 10 |
| rdsLogClear() .....                        | 10 |
| Profile functions .....                    | 11 |
| Struct rdsProfileInfo .....                | 11 |
| Struct rdsProfileInfo1 .....               | 14 |
| Struct rds_firmware_t .....                | 15 |
| Struct rds_application_t .....             | 16 |
| Struct rds_folder_t .....                  | 17 |
| Struct rds_fw_limit_t .....                | 18 |
| Callback notify_profile_data .....         | 18 |
| Callback notify_profile1_data .....        | 19 |
| Callback notify_profile_deleted .....      | 19 |
| rdsRegisterProfileNotification() .....     | 20 |
| rdsRegisterProfile1Notification() .....    | 20 |
| rdsProfileLoad() .....                     | 21 |
| rdsProfileCreate() .....                   | 21 |
| rdsProfile1Create() .....                  | 22 |
| UdsProfileDelete() .....                   | 23 |
| rdsProfileGetCount() .....                 | 23 |
| rdsProfileEdit() .....                     | 24 |
| rdsProfile1Edit() .....                    | 25 |
| Unit functions .....                       | 26 |
| Struct rdsUnitInfo .....                   | 26 |
| Struct rdsImportUnit .....                 | 28 |
| Callback notify_unit_data .....            | 28 |

|                                     |    |
|-------------------------------------|----|
| Callback notify_unit_deleted.....   | 29 |
| rdsRegisterUnitNotification().....  | 29 |
| rdsUnitLoad().....                  | 29 |
| rdsUnitCreate().....                | 30 |
| rdsUnitDelete() .....               | 30 |
| rdsUnitEdit() .....                 | 31 |
| rdsUnitImport().....                | 32 |
| rdsUnitReset() .....                | 32 |
| rdsUnitRefresh() .....              | 33 |
| rdsUnitTransferForceApp() .....     | 33 |
| rdsUnitTransferForceFile().....     | 34 |
| rdsUnitTransferForceFw().....       | 34 |
| rdsUnitTransferForceRuntime() ..... | 35 |
| File functions.....                 | 36 |
| Callback cbFilename .....           | 36 |
| rdsGetFilelist() .....              | 37 |

## Introduction

This document discusses the library RDSAPI.DLL, which enables you to control the RTCU Deployment Server (RDS) from your own code. The library provides functions to manage profiles and units, providing the same functionality as the RTCU Deployment Server Manager. The library connects to the RTCU Deployment Server through the RTCU Gateway 2 which is connected to the RTCU units.

## Graphical illustration of the system

## Contents of package

The package this document is part of, contains the following:

- RTCU Deployment Server API.pdf                      This document
- \Library\    The .H, .LIB and .DLL files needed for the RDSAPI library.

In order to use this library, you will need to use Microsoft Visual Studio C++ 2005 or newer.

## Functions in the RDSAPI.DLL library

### Server Functions

#### rdsConnect()

**Synopsis** int rdsConnect(const char\* gw\_addr, const unsigned short gw\_port, const char\* gw\_key, const char\* rds\_key)

**Description** Opens the connection to the gateway and logs on to the RDS. The connection is opened and managed asynchronously. Use the rdsGetStatus function to determine when the RDSAPI library is connected.  
The connection can be closed again by calling rdsDisconnect().

#### Input

|         |                                |
|---------|--------------------------------|
| gw_addr | The IP address of the gateway. |
| gw_port | The IP port of the gateway.    |
| gw_key  | The logon key for the gateway. |
| rds_key | The logon key for the RDS.     |

#### Returns

- 2 Connection is already open
- 1 Illegal parameter
- 0 Success

#### rdsDisconnect()

**Synopsis** void rdsDisconnect(void)

**Description** Closes the connection to the gateway and the RDS. The connection is closed when the function returns.

**Returns** None.

#### rdsGetStatus()

**Synopsis** int rdsGetStatus(void)

**Description** Gets the status of the connection to the server.

**Returns** One of the following values:

- 15 Unsupported RDS.
- 14 Incorrect RDS login password.
- 13 All monitor slots on the server is in use.

- 12 RDS not found.
- 4 Connected to RDS.
- 3 Connecting to RDS.
- 2 Connected to gateway.
- 1 Connecting to gateway.
- 0 Not connected.

## rdsGetVersion()

**Synopsis**                   int rdsGetVersion(void)

**Description**             Get the version of the RDS API library

**Returns**                  The version of the RDSAPI library scaled by 100.

## rdsGetServerVersion()

**Synopsis**                   int rdsGetServerVersion(void)

**Description**             Gets the version of the connected RDS.

**Returns**                  The version number of the server multiplied by 100, or -1 if the RDS Server is not connected.

## Callback cbStatus

**Synopsis**                   typedef void (\*cbStatus)(void\* arg, const char\* Name, const char Status);

**Description**             Is used when requesting the status of the gateway connections of the server.

### Input

|        |   |
|--------|---|
| arg    | A user supplied argument.                             |
| Name   | The name of the gateway. (Zero terminated ASCII text) |
| Status | Zero = Not connected, <> zero = Connected             |

**Returns**                  None.

## rdsGetServerStatus()

**Synopsis**                   int rdsGetServerStatus(void\* arg, cbStatus cbFunc);

**Description** Request the server connection status.

**Input**

|        |  |
|--------|--|
| arg    | A user supplied argument, that is passed with the callback.                                    |
| cbFunc | Pointer to the function that the RDSAPI will call with the connection status for each gateway. |

**Returns**

Number of gateways the RDS connects to.

- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.



## Log functions

### Callback notify\_log\_data

**Synopsis** typedef void (\*notify\_log\_data)(void\* arg, const time\_t timestamp, const char\* text);

**Description** Is called when a log entry is received.

Input

|           |  |
|-----------|--|
| arg       | A user supplied argument.                                |
| timestamp | The timestamp for the log entry.                         |
| text      | The text for the log entry. (Zero terminated ASCII text) |

**Returns** None.

### Callback notify\_log\_clear

**Synopsis** typedef void (\*notify\_log\_clear)(void\* arg);

**Description** Is called when the log is cleared.

Input

|     |                           |
|-----|---------------------------|
| arg | A user supplied argument. |
|-----|---------------------------|

**Returns** None.

### rdsRegisterLogNotification()

**Synopsis** void rdsRegisterLogNotification(void\* arg, notify\_log\_data data, notify\_log\_clear clear)

**Description** Register callback functions with RDSAPI library, which is then called to notify about log changes.

Input

|       |  |
|-------|--|
| arg   | A user supplied argument, that is included with the notifications.                         |
| data  | The function to notify when log data is received. (The received log data will be included) |
| clear | The function to notify when log is cleared.  |

**Returns** None.

## **rdsLogLoad()**

**Synopsis**                   int rdsLogLoad(void)

**Description**             Request the RDS log. The callback data registered in rdsRegisterLogNotification() will be called for each entry in the log.

**Returns**

- 1 General error.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## **rdsLogClear()**

**Synopsis**                   int rdsLogClear(void)

**Description**             Clears the RDS log.

**Returns**

- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## Profile functions

### Struct rdsProfileInfo

#### Synopsis

```
typedef struct rdsProfileInfo {
    long        id_number;
    char        name[65];
    char        password[21];
    int         flags;
    long        time_start;
    long        time_stop;

    char        app_name[16];
    short       app_ver;
    char        app_file[191];

    short       fw_tgt;
    int         fw_system;
    int         fw_runtime;
    int         fw_monitor;
    char        fw_file[191];

    char        file_file[191];
    char        file_dst[61];

    int         app_newer;
    int         app_older;
    rds_fw_limit_t fw_newer;
    rds_fw_limit_t fw_older;
} rdsProfileInfo;
```

**Description**                Stores the information about a profile

#### Fields

|            |   |
|------------|---|
| id_number  | id number of the profile. Read only.  |
| name       | The name of the profile   |
| password   | The password for the units.   |
| flags      | Flags for the profile. See table below for possible values.   |
| time_start | The start of the transfer time interval. Seconds since midnight. Use -1 to disable the time limitation. |
| time_stop  | The end of the transfer time interval. Seconds since midnight. Use -1 to disable the time limitation.   |
| app_name   | The name of the application.  |
| app_ver    | The version number of the application (scaled by 100).  |
| app_file   | The file name of the application.   |

|            |   |
|------------|---|
| fw_tgt     | The target ID for the firmware                          |
| fw_system  | The firmware version or the system version (NX32L only) |
| fw_runtime | The runtime version. (NX32L only)                       |
| fw_monitor | The monitor version. (NX32L only)                       |
| fw_file    | The file name of the firmware                           |
| file_file  | File name of file to transfer                           |
| file_dst   | Destination drive for file, e.g. "A:"                   |
| app_newer  | Application policy: Only newer than                     |
| app_older  | Application policy: Only older than                     |
| fw_newer   | Firmware policy: Only newer than                        |
| fw_older   | Firmware policy: Only older than                        |

**Target ID**

|               |     |                     |
|---------------|-----|---------------------|
| TARGET_A9     | 9   | A9 / A9i.           |
| TARGET_M10    | 10  | M10.                |
| TARGET_M11    | 11  | M11.                |
| TARGET_MX2    | 102 | MX2 series.         |
| TARGET_DX4MK2 | 103 | DX4i mk2.           |
| TARGET_DX4    | 104 | DX4 series.         |
| TARGET_CX1    | 105 | CX1 series.         |
| TARGET_SX1    | 106 | SX1 series.         |
| TARGET_AX9    | 109 | AX9 series.         |
| TARGET_MX2T   | 122 | MX2 turbo / encore. |
| TARGET_AX9T   | 129 | AX9 turbo / encore. |
| TARGET_MX2SOM | 192 | MX2 SOM.            |
| TARGET_EGATE  | 201 | eGate.              |
| TARGET_NX200  | 202 | NX-200.             |
| TARGET_NX400  | 204 | NX-400.             |
| TARGET_NX900  | 209 | NX-900.             |
| TARGET_LTGW   | 211 | LT-GW.              |
| TARGET_LTGW2  | 212 | LT-GW2.             |
| TARGET_LX100  | 301 | LX-100.             |
| TARGET_LX2    | 302 | LX2.                |
| TARGET_LX4    | 304 | LX4.                |
| TARGET_LX5    | 305 | LX5.                |

|                   |     |              |
|-------------------|-----|--------------|
| TARGET_LX5_ULTRA  | 315 | LX5 Ultra.   |
| TARGET_LX5_CORE   | 316 | LX5 Core.    |
| TARGET_LX9_PRO    | 309 | LX9 Pro.     |
| TARGET_LX9_ECO    | 319 | LX9 Eco.     |
| TARGET_LTGW_BASIC | 331 | LT-GW Basic. |
| TARGET_EGATEV2    | 335 | eGate V2.    |
| TARGET_ZHAGA      | 340 | Zhaga.       |

**Profile flags**

|                        |            |  |
|------------------------|------------|--|
| PFLAG_AUTORESET        | 0x00000001 | When set, the RDS will reset the RTCU units when a transfer is completed.  |
| PFLAG_SETPASSWORD      | 0x00000002 | The password in the RTCU units will be set to the value supplied in password.  |
| PFLAG_DEACTIVATE       | 0x00000004 | If set, units that use this profile will not be upgraded.  |
| PFLAG_APP_ENABLE       | 0x00000008 | The update includes an application.  |
| PFLAG_FW_ENABLE        | 0x00000010 | The update includes a firmware   |
| PFLAG_FORCEHALT        | 0x00000020 | The application is halted during the transfer.   |
| PFLAG_TARGETDEFAULT    | 0x00000040 | The profile is the default for the target.   |
| PFLAG_FILE_ENABLE      | 0x00000080 | The update includes a file.  |
| PFLAG_NO_DOWNGRADE_APP | 0x00000100 | When set, the RDS will not transfer the application if the version in the device is newer than the version in the profile.                               |
| PFLAG_NO_DOWNGRADE_FW  | 0x00000200 | When set, the RDS will not transfer the firmware if the version in the device is newer than the version in the profile.                                  |
| PFLAG_APP_NEWER        | 0x00000400 | When set, the RDS will not transfer the application if the version in the device is older that or equal to the version given in the app_newer parameter. |
| PFLAG_APP_OLDER        | 0x00000800 | When set, the RDS will not transfer the application if the version in the device is newer that or equal to the version given in the app_older parameter. |
| PFLAG_FW_NEWER         | 0x00001000 | When set, the RDS will not transfer the firmware if the version in the device is older that or equal to the version given in the fw_newer parameter.     |
| PFLAG_FW_OLDER         | 0x00002000 | When set, the RDS will not transfer the firmware if the version in the device is newer that or equal to the version given in the fw_older parameter.     |

|                  |            |  |
|------------------|------------|--|
| PFLAG_APP_INFILE | 0x40000000 | This flag is used by the Monitor Tool to indicate that the user cannot type in the application name and version. |
|------------------|------------|--|

## Struct rdsProfileInfo1

### Synopsis

```
typedef struct rdsProfileInfo1 {
    long            size;
    long            id_number;
    char            name[65];
    char            password[21];
    int             flags;
    long            time_start;
    long            time_stop;

    char            file_file[191];
    char            file_dst[61];

    rds_firmware_t  system;
    rds_firmware_t  runtime;
    rds_firmware_t  monitor;
    rds_application_t app;
    rds_folder_t    folder
} rdsProfileInfo;
```

**Description**                Stores the information about a profile

### Fields

|            |   |
|------------|---|
| size       | The size of the structure in bytes.   |
| id_number  | id number of the profile. Read only.  |
| name       | The name of the profile   |
| password   | The password for the units.   |
| flags      | Flags for the profile. See table below for possible values.   |
| time_start | The start of the transfer time interval. Seconds since midnight. Use -1 to disable the time limitation. |
| time_stop  | The end of the transfer time interval. Seconds since midnight. Use -1 to disable the time limitation.   |
| app        | The application information.  |
| folder     | The directory synchronization information. (NX32L only)   |
| monitor    | The monitor firmware information (NX32L only)   |
| runtime    | The runtime firmware information. (NX32L only)  |
| system     | The system firmware information.  |
| file_file  | The name of the file to transfer  |
| file_dst   | Destination drive for file, e.g. "A:"   |

## Struct rds\_firmware\_t

### Synopsis

```
struct rds_firmware_t {
    int         flags;
    int         target;
    int         system;
    int         runtime;
    int         monitor;
    rds_fw_limit_t newer;
    rds_fw_limit_t older;
    char        file[191];
};
```

**Description** Stores the information about a firmware update.

### Fields

|         |  |
|---------|--|
| flags   | Flags for the firmware. See table below for possible values. |
| target  | The target ID for the firmware                               |
| system  | The firmware version or the system version (NX32L only)      |
| runtime | The runtime version. (NX32L only)                            |
| monitor | The monitor version. (NX32L only)                            |
| newer   | Policy: Only newer than                                      |
| older   | Policy: Only older than                                      |
| file    | The file name of the firmware                                |

### Firmware flags

|                |            |  |
|----------------|------------|--|
| FLAG_ENABLE    | 0x00000001 | Enable the firmware update   |
| FLAG_DOWNGRADE | 0x00000002 | When set, the RDS will transfer the firmware if the version in the device is newer than the version in the profile.                                  |
| FLAG_NEWER     | 0x00000004 | When set, the RDS will not transfer the firmware if the version in the device is older than or equal to the version given in the fw_newer parameter. |
| FLAG_OLDER     | 0x00000008 | When set, the RDS will not transfer the firmware if the version in the device is newer than or equal to the version given in the fw_older parameter. |



## Struct rds\_application\_t

### Synopsis

```
struct rds_firmware_t {
    int      flags;
    int      newer;
    int      older;
    int      version;
    char     file[191];
    char     name[16];
};
```

**Description** Stores the information about the application update.

### Fields

|         |   |
|---------|---|
| flags   | Flags for the application. See table below for possible values. |
| newer   | Policy: Only newer than   |
| older   | Policy: Only older than   |
| version | The version number of the application (scaled by 100).          |
| file    | The file name of the firmware                                   |
| name    | The name of the application.                                    |

### Application flags

|                |            |  |
|----------------|------------|--|
| FLAG_ENABLE    | 0x00000001 | Enable the application update  |
| FLAG_DOWNGRADE | 0x00000002 | When set, the RDS will transfer the application if the version in the device is newer than the version in the profile.                                   |
| FLAG_NEWER     | 0x00000004 | When set, the RDS will not transfer the firmware if the version in the device is older than or equal to the version given in the fw_newer parameter.     |
| FLAG_OLDER     | 0x00000008 | When set, the RDS will not transfer the application if the version in the device is newer than or equal to the version given in the app_older parameter. |

## Struct rds\_folder\_t

### Synopsis

```
struct rds_folder_t {  
    int      flags;  
    int      media;  
    char     folder[191];  
};
```

**Description** Stores the information about the directory synchronization.

### Fields

|        |   |
|--------|---|
| flags  | Flags for the directory synchronization. See table below for possible values. |
| media  | Destination media for synchronization, e.g. 0 for SD-CARD.                    |
| folder | The directory which will be synchronized.                                     |

### Directory synchronization flags

|             |            |   |
|-------------|------------|---|
| FLAG_ENABLE | 0x00000001 | Enable the directory synchronization  |
| FLAG_DELETE | 0x00000010 | When set, the RDS will delete the contents of the directory after it is transferred to the devices. |

## Struct rds\_fw\_limit\_t

### Synopsis

```
struct rds_fw_limit_t {  
    int          system;  
    int          runtime;  
};
```

**Description** Stores the version for the firmware 'Only newer than' and 'Only older than' policy.

### Fields

|         |   |
|---------|---|
| system  | The firmware version or the system version (NX32L only) |
| runtime | The runtime version. (only valid for NX32L devices)     |

## Callback notify\_profile\_data

**Synopsis** typedef void (\*notify\_profile\_data)(void\* arg, const rdsProfileInfo\* item);

**Description** Is called when a profile is changed.

### Input

|      |  |
|------|--|
| arg  | A user supplied argument.                        |
| item | Pointer to a structure holding the profile data. |

**Returns** None.

## Callback notify\_profile1\_data

**Synopsis**                    typedef void (\*notify\_profile1\_data)(void\* arg, const rdsProfileInfo1\* item);

**Description**              Is called when a profile is changed.

### Input

|      |  |
|------|--|
| arg  | A user supplied argument.                        |
| item | Pointer to a structure holding the profile data. |

**Returns**                    None.

## Callback notify\_profile\_deleted

**Synopsis**                    typedef void (\*notify\_profile\_deleted)(void\* arg, const long ID);

**Description**              Is called when a profile is deleted.

### Input

|     |   |
|-----|---|
| arg | A user supplied argument.                     |
| ID  | The ID of the profile that have been deleted. |

**Returns**                    None.

## rdsRegisterProfileNotification()

**Synopsis** void rdsRegisterProfileNotification(void\* arg, notify\_profile\_data data, notify\_profile\_deleted deleted);

**Description** Register callback functions with RDSAPI library, which is then called to notify about profile changes.

Input

|         |  |
|---------|--|
| arg     | A user supplied argument, that is included with the notifications.                                 |
| data    | The function to notify when profile data is received. (The received profile data will be included) |
| deleted | The function to notify when a profile is deleted.  |

**Returns** None

## rdsRegisterProfile1Notification()

**Synopsis** void rdsRegisterProfile1Notification(notify\_profile1\_data data);

**Description** Register callback function with RDSAPI library, which is then called to notify about profile changes.

Input

|      |  |
|------|--|
| data | The function to notify when profile data is received. (The received profile data will be included) |
|------|--|

**Returns** None

## rdsProfileLoad()

**Synopsis**                   int rdsProfileLoad(void)

**Description**             Request the data of all the profiles on the RDS. The callback data registered in rdsRegisterProfileNotification() will be called for each profile.

**Returns**

- 1 General error
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsProfileCreate()

**Synopsis**                   int rdsProfileCreate(rdsProfileInfo\* Item);

**Description**             Create a new profile on the RDS.

**Input**

|      |   |
|------|---|
| Item | Pointer to the structure that holds the profile data. |
|------|---|

**Returns**

- 9 No Application, Firmware and File is selected.
- 8 Name is missing.
- 7 Invalid time interval.
- 6 Invalid firmware parameter.
- 5 Invalid application parameter.
- 4 Invalid file parameter
- 3 Other profile is set as default for target.
- 2 Other profile with the same name.
- 1 General error
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsProfile1Create()

**Synopsis**                   int rdsProfile1Create(rdsProfileInfo1\* Item);

**Description**             Create a new profile on the RDS.

### Input

|      |   |
|------|---|
| Item | Pointer to the structure that holds the profile data. |
|------|---|

### Returns

- 9 No Application, Firmware and File is selected.
- 8 Name is missing.
- 7 Invalid time interval.
- 6 Invalid firmware parameter.
- 5 Invalid application parameter.
- 4 Invalid file parameter
- 3 Other profile is set as default for target.
- 2 Other profile with the same name.
- 1 General error
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## UdsProfileDelete()

**Synopsis**                   int rdsProfileDelete(long ID);

**Description**             Delete a profile from the RDS.

### Input

|    |                        |
|----|------------------------|
| ID | The ID of the profile. |
|----|------------------------|

### Returns

- 2 Profile is in use.
- 1 Profile is not found.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsProfileGetCount()

**Synopsis**                   int rdsProfileGetCount(void);

**Description**             Request the number of profiles on the RDS.

**Returns**                  The number of profiles on the RDS Server, or 0 in case of an error.



## rdsProfileEdit()

**Synopsis**                   int rdsProfileEdit(rdsProfileInfo\* Item);

**Description**             Edit the data of a profile on the RDS.

### Input

|      |   |
|------|---|
| Item | Pointer to the structure that holds the profile data. |
|------|---|

### Returns

- 10 No Application, Firmware and File is selected.
- 9 Name is missing.
- 8 Invalid time interval.
- 7 Invalid firmware parameter.
- 6 Invalid application parameter.
- 5 Invalid file parameter.
- 4 Other profile is set as default for target.
- 3 Other profile with the same name
- 1 Profile is not found.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsProfile1Edit()

**Synopsis**                   int rdsProfile1Edit(rdsProfileInfo1\* Item);

**Description**            Edit the data of a profile on the RDS.

### Input

|      |   |
|------|---|
| Item | Pointer to the structure that holds the profile data. |
|------|---|

### Returns

- 10 No Application, Firmware and File is selected.
- 9 Name is missing.
- 8 Invalid time interval.
- 7 Invalid firmware parameter.
- 6 Invalid application parameter.
- 5 Invalid file parameter.
- 4 Other profile is set as default for target.
- 3 Other profile with the same name
- 1 Profile is not found.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## Unit functions

### Struct rdsUnitInfo

#### Synopsis

```
typedef struct rdsUnitInfo {
    long        serial_number;
    long        profile_id;
    char        password[21];
    int         flags;
    char        app_name[16];
    short       app_ver;
    short       fw_tgt;
    int         fw_system;
    int         fw_runtime;
    int         fw_monitor;
    char        progress;
    char        gateway;
    short       status;
    time_t      timestamp;
    char        comment[101];
} rdsUnitInfo;
```

**Description**                Stores the information about a unit

#### Fields

|               |   |
|---------------|---|
| serial_number | serial number of the unit   |
| profile_id    | id number of the profile  |
| password      | The password for the unit.  |
| flags         | Flags for the unit. See table below for values.                             |
| app_name      | application name. (15 chars + zero terminator)                              |
| app_ver       | application version (scaled by 100)   |
| fw_tgt        | Target ID of the device.  |
| fw_system     | firmware version (scaled by 100) or the system version (NX32L)              |
| fw_runtime    | The runtime version.(NX32L)   |
| fw_monitor    | The monitor version.(NX32L)   |
| progress      | transfer progress in percent (%)  |
| gateway       | index of the Gateway that the unit is connected to.                         |
| status        | Status of the unit. See the values in the table below.                      |
| timestamp     | Time stamp of the last time the unit connected and was completely upgraded. |
| comment       | User defined comment.   |

**Unit status values**

|                                |    |  |
|--------------------------------|----|--|
| USTATUS_NO_INFO                | 0  | No information has been received from the RTCU unit yet.   |
| USTATUS_NOT_CONNECTED          | 1  | RTCU unit is currently not connected to the RDS.   |
| USTATUS_UP_TO_DATE             | 2  | RTCU unit is up to date. The timestamp is the time for the last reset.   |
| USTATUS_DEACTIVATED_PROFILE    | 3  | The profile in use has been deactivated.   |
| USTATUS_DEACTIVATED            | 4  | The unit has been deactivated.   |
| USTATUS_TRANSFER_IN_QUEUE      | 5  | RTCU unit is not up to date, and a transfer has been queued  |
| USTATUS_TRANSFER_APP           | 6  | RDS is transferring the application to the RTCU unit.  |
| USTATUS_TRANSFER_FW            | 7  | RDS is transferring the firmware to the RTCU unit.   |
| USTATUS_WAIT_FOR_RESET         | 8  | Upload has been completed, and the RDS is waiting for the RTCU unit to reset.  |
| USTATUS_WAIT_FOR_TIME          | 9  | Waiting for upgrade time interval to arrive  |
| USTATUS_TRANSFER_FILE          | 10 | RDS is transferring the generic file to the RTCU unit.   |
| USTATUS_TRANSFER_SYNC          | 11 | RDS is transferring the directory synchronization contents to the RTCU unit.   |
| USTATUS_TRANSFER_SYNC_WAIT     | 12 | Waiting for RTCU device to report that the folder is updated.  |
| USTATUS_ERROR_APP              | 30 | The RTCU is not programmable or does not support EIS.  |
| USTATUS_ERROR_FW               | 31 | Firmware is not targeted for the RTCU unit type.   |
| USTATUS_ERROR_FILE_APP         | 32 | RDS could not find the application file.   |
| USTATUS_ERROR_FILE_APP_INVALID | 33 | File is not a valid application file.  |
| USTATUS_ERROR_FILE_FW          | 34 | RDS could not find the firmware file.  |
| USTATUS_ERROR_FILE_FW_INVALID  | 35 | File is not a valid firmware file.   |
| USTATUS_ERROR_PASSWORD         | 36 | The password in the profile or unit is not identical to the password in the RTCU unit.   |
| USTATUS_ERROR_TRANSFER         | 37 | Version mismatch. Check the profile for typing errors and make sure the application sets the application name and version correctly. |
| USTATUS_ERROR_FILE_TRANSFER    | 38 | RDS could not transfer the generic file.   |
| USTATUS_ERROR_FAILED           | 39 | The device repeatedly failed to be updated, so the update has been stopped.  |

**Unit flags**

|                  |            |  |
|------------------|------------|--|
| UFLAG_DEACTIVATE | 0x00010000 | If set, the unit will not be upgraded. |
|------------------|------------|--|

## Struct rdsImportUnit

### Synopsis

```
typedef struct rdsImportUnit {
    long        serial_number;
    long        profile_id;
    char        flag;
    char        comment[101];
} rdsImportUnit;
```

**Description**            Stores information about a unit to import

### Fields

|               |  |
|---------------|--|
| serial_number | serial number of the unit to import            |
| profile_id    | id number of the profile for the imported unit |
| flag          | Set to 1 to enable the unit, 0 to disable it.  |
| comment       | User defined comment.                          |

## Callback notify\_unit\_data

**Synopsis**                typedef void (\*notify\_unit\_data)(void\* arg, const rdsUnitInfo\* item);

**Description**            Is called when a unit is changed.

### Input

|      |   |
|------|---|
| arg  | A user supplied argument.                     |
| item | Pointer to a structure holding the unit data. |

**Returns**                None.

## Callback notify\_unit\_deleted

**Synopsis** typedef void (\*notify\_unit\_deleted)(void\* arg, const long ID);

**Description** Is called when a unit is deleted.

### Input

|     |  |
|-----|--|
| arg | A user supplied argument.                  |
| ID  | The ID of the unit that have been deleted. |

**Returns** None.

## rdsRegisterUnitNotification()

**Synopsis** void rdsRegisterUnitNotification(void\* arg, notify\_unit\_data data, notify\_unit\_deleted deleted);

**Description** Register callback functions with RDSAPI library, which is then called to notify about unit changes.

### Input

|         |  |
|---------|--|
| arg     | A user supplied argument, that is included with the notifications.                           |
| data    | The function to notify when unit data is received. (The received unit data will be included) |
| deleted | The function to notify when a unit is deleted.   |

**Returns** None.

## rdsUnitLoad()

**Synopsis** int rdsUnitLoad(void)

**Description** Request the data of all the units on the RDS The callback data registered in rdsRegisterUnitNotification() will be called for each unit.

### Returns

- 1 General error.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitCreate()

**Synopsis** int rdsUnitCreate(const long ID, const long ProfileID, const int flags, const char\* Password, const char\* Comment);

**Description** Create a new unit on the RDS.

### Input

|           |  |
|-----------|--|
| ID        | The ID of the unit.                                    |
| ProfileID | The ID of the profile used.                            |
| flags     | The flags of the unit.                                 |
| Password  | Zero terminated ASCII string with the unit password.   |
| Comment   | Zero terminated ASCII string with comments about unit. |

### Returns

- 3 Invalid parameter.
- 2 Other unit exists.
- 1 General error.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitDelete()

**Synopsis** int rdsUnitDelete(long ID);

**Description** Delete a unit from the RDS.

### Input

|    |                     |
|----|---------------------|
| ID | The ID of the unit. |
|----|---------------------|

### Returns

- 1 Unit is not found
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitEdit()

**Synopsis**                   int rdsUnitEdit(const long ID, const long ProfileID, const int flags, const char\* Password, const char\* Comment);

**Description**            Edit a unit on the RDS.

### Input

|           |  |
|-----------|--|
| ID        | The ID of the unit.                                    |
| ProfileID | The ID of the profile used.                            |
| flags     | The flags of the unit.                                 |
| Password  | Zero terminated ASCII string with the unit password.   |
| Comment   | Zero terminated ASCII string with comments about unit. |

### Returns

- 2 Invalid parameter.
- 1 Unit is not found.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.



## rdsUnitImport()

**Synopsis** `int rdsUnitImport(const char flags, const int count, rdsImportUnit* items);`

**Description** Import a number of units to the RDS.

### Input

|       |  |   |   |
|-------|--|---|---|
| flags | Currently have the following flag:                               |   |   |
|       | RDS_IMPORT_OVERWRITE   | 1 | If set, the imported units will overwrite existing units. |
| count | The number of units in the array.                                |   |   |
| items | Pointer to an array of structures that hold the units to import. |   |   |

### Returns

- 3 Invalid parameter.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitReset()

**Synopsis** `int rdsUnitReset(const int count, const long* units);`

**Description** Reset a number of units on the RDS.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitRefresh()

**Synopsis**                    `int rdsUnitRefresh(const int count, const long* units);`

**Description**              Refresh the information for a number of units on the RDS. Only changed units will cause notifications.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitTransferForceApp()

**Synopsis**                    `int rdsUnitTransferForceApp(const int count, const long* units);`

**Description**              Force an application transfer for a number of units on the RDS.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitTransferForceFile()

**Synopsis**                   int rdsUnitTransferForceFile(const int count, const long\* units);

**Description**            Force the transfer of a generic file for a number of units on the RDS.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitTransferForceFw()

**Synopsis**                   int rdsUnitTransferForceFw(const int count, const long\* units);

**Description**            Force a firmware transfer for a number of units on the RDS.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## rdsUnitTransferForceRuntime()

**Synopsis**                   int rdsUnitTransferForceRuntime(const int count, const long\* units);

**Description**             Force a runtime firmware transfer for a number of units on the RDS.  
Only available for NX32L.

### Input

|       |   |
|-------|---|
| count | The number of units in the array.           |
| units | Pointer to an array that hold the unit ids. |

### Returns

- 1 No units in array.
- 0 Success.
- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.

## File functions

### Callback cbFilename

**Synopsis** typedef void (\*cbFilename)(void\* arg, const char\* Filename, const int Target, const int System, const int Runtime, const int Monitor, const char\* Name);

**Description** Is called when requesting a list of files. (Both application files, Firmware files and Folders)

#### Input

|          |   |
|----------|---|
| arg      | A user supplied argument.   |
| Filename | The name of the file or directory. (Zero terminated ASCII text)   |
| Target   | The target id of the firmware. -1 is returned for Application and Folder.                                       |
| System   | The version of the file, scaled by 100. -1 is returned for Folder.<br>The system version for NX32L devices.     |
| Runtime  | The runtime version. -1 for folders and 0 if not a NX32L device.  |
| Monitor  | The monitor version. -1 for folders and 0 if not a NX32L device.  |
| Name     | The name of the application. (Zero terminated ASCII text.) An empty string is returned for Firmware and Folder. |

**Returns** None.

## rdsGetFilelist()

**Synopsis**                   int rdsGetFilelist(void\* arg, cbFilename cbFunc, const int type, const char\* Folder);

**Description**             Retrieves a list of files from the RDS.

### Input

|        |   |
|--------|---|
| arg    | A user supplied argument, that is passed with the callback.   |
| cbFunc | Pointer to the function that the RDSAPI will call with the files found in the Folder                      |
| type   | The type of file:<br>0: Application<br>1: Firmware<br>2: Generic file<br>3: Directory for synchronization |
| Folder | The path of the sub-directory to query.   |

### Returns

Number of files in the folder

- 1 Not connected to gateway.
- 2 RDS not connected to gateway.
- 3 Communication error.