>DIGI INTERNATIONAL

Adding new RCI configuration elements Rev A

[This document describes the process of adding new RCI configuration elements.]

Table of Contents

Overview

This document describes the process of adding new RCI configuration elements that allow access to the Kontron device through the Device Cloud. This process consists of the following;

1. Defining how the information will be displayed by the Device Cloud.
2. Uploading display information to the Device Cloud.
3. Modifying the callback definitions.
4. Modifying the callback routines.
5. Build and run the new image in the Wind River Workbench.
6. Viewing the menu in Device Manager.

A working sample of adding RCI configuration elements for Network Time Protocol (NTP) is provided. The RCI files that will be modified are in the /home/wruser/WindRiver/workspace/connector_rci_files folder. Any changes made to the RCI files will affect all the projects because this is a shared folder.

Java Requirements

Oracle Java 1.6 update 22 or greater is recommended.

Defining how the information will be displayed by the Device Cloud

The file "config.rci" is located on the Linux PC running "Live USB" in the /home/wruser/connector directory. The file "config.rci" contains all of the information on how items will be displayed by the Device Cloud. This information will be used to create the files "rci_config.c" and "rci_config.h" that are needed to write the code for the callback routines. Information on "config.rci" can be found in the Connector Users Guide on the Desktop under the **"**Remote Configuration File**"** section in the "Remote Configuration" topic.

Example: Add new NTP configuration setting information to **"**config.rci**"**.

```
# NTP Options
 group setting ntp "NTP"
     element server "Server" type string max 255
     element port "Port" type string max 255
```

Uploading display information to the Device Cloud

There is a provisioning icon on the desktop of the host computer. This will invoke a script which will provision the device. Provisioning the device pushes up the new configuration information to the Device Cloud. Double click on the provisioning icon on the desktop and enter the information requested.

Modifying the callback definitions

You will need to modify "remote_config_cb.c" when adding new RCI configuration elements. Currently only "setting" and "state" menu types are supported, but other menu types can be defined. When adding a new "setting" or "state", table entries must be added for the "init", "set", "get" and "end" functions in the "remote_setting_table[]" or the "remote_state_table[]". These functions are defined in the callback routines. Function prototypes are added to "remote_config_cb.h".

Example: Add new NTP configuration setting definitions.

Step 1: Add "ntp_group_init", "ntp_group_set", "ntp_group_end", "ntp_group_get" and "ntp_group_cancel" definitions to the end of the "remote_setting_table[]" in "remote_config_cb.c".

```
{ntp_group_init, ntp_group_set, ntp_group_get, ntp_group_end, ntp_group_cancel},
```

Step 2: Add these lines to **"**remote_config_cb.h**"**.

```
extern connector_callback_status_t ntp_group_init(connector_remote_config_t * const remote_config);
extern connector_callback_status_t ntp_group_get(connector_remote_config_t * const remote_config);
extern connector_callback_status_t ntp_group_set(connector_remote_config_t * const remote_config);
extern connector_callback_status_t ntp_group_end(connector_remote_config_t * const remote_config);
extern void ntp_group_cancel(connector_remote_config_cancel_t * const remote_config);
```

Modifying the callback routines

The file "rci_config.h" specifies all of the items that will be accessed by the "group_init", "group_get", "group_set" and "group_end" functions defined in "remote_config_cb.c" and "remote_config_cb.h". Currently the functions are in “access_point.c”, "cellular.c", "ethernet.c", “gps.c”, "wifi_client.c” and "system.c". Use these files as a guide when writing your functions. The "group_init", "group_get", "group_set" and "group_end" functions are designed to do the following;

"**group_init_function**" - This function is responsible for mallocing the structure to hold the " ntp_config_t " and pointing this memory to the "session_ptr->group_context". It is also responsible for reading the current values of the variables into the "ntp_config_t " structure.

"**group_get_function**" - This function is responsible for responding to queries from Device Manager with variables from the internal database.

"**group_set_function**" - This function accepts setting data from the Device Cloud and enters the variables into the appropriate place in the "ntp_config_t" structure.

"**group_end_function**" - This function is responsible for writing all of the variables in the "ntp_config_t" structure to the database.

All variables are accessed using UCI (Unified Configuration Interface) configuration. More information on UCI can be found at http://wiki.openwrt.org/doc/uci. UCI is a small utility intended to centralize the device configuration. UCI is the successor of the NVRAM based configuration. The UCI configuration files are located in the /etc/config directory on the Kontron device. For example, the "wireless" configuration file is called "wireless". The Ethernet and Cellular configuration file is called "network". The "NTP" file is called "ntpclient".

Example: Add new NTP configuration setting in a new file "ntp.c".

Step 1: Define "ntp_config_data_t".

"ntp_config_data_t" will contain all of the NTP configuration objects that can be accessed by the Device Cloud. This is an example of the include files needed to compile the file and the NTP configuration structure.

```
#include <stdlib.h>
#include <net/if.h>
#include "connector.h"
#include "remote_config_cb.h"
#include "connector_db.h"
#include "rci_config.h"
```

```
typedef  struct   {
      char  server[MAX_STRLEN];
      char  port[MAX_STRLEN];
} ntp_config_data_t;
```

Step 2: Add "ntp_group_init".

```
connector_callback_status_t ntp_group_init(connector_remote_config_t * const remote_config)
{
       remote_group_session_t * const session_ptr = remote_config->user_context;
       connector_callback_status_t status = connector_callback_abort;
       ntp_config_data_t * ntp_ptr;
       char str[MAX_STRLEN];

       ASSERT(session_ptr != NULL);

       session_ptr->group_context = malloc(sizeof(ntp_config_data_t));
       if (session_ptr->group_context == NULL)
       {
          remote_config->error_id = connector_global_error_memory_fail;
          goto done;
       }

       memset(session_ptr->group_context, 0, sizeof(ntp_config_data_t));

       ntp_ptr = session_ptr->group_context;

       if (uci_read_config_var("ntpclient", "ntp_server",  str,  "general",  sizeof  str) < 0)
       {
          memset (str, 0, sizeof str);
       }

       /* Need to remove \n from string to avoid RCI error */
       if (str[strlen(str) - 1] == 0x0a)
          str[strlen(str) - 1] = 0;

       strncpy(ntp_ptr->server, str, sizeof  ntp_ptr->server);

       if (uci_read_config_var("ntpclient", "ntp_port",  str, "general",  sizeof  str) < 0)
       {
          memset (str, 0, sizeof str);
       }

       /* Need to remove \n from string to avoid RCI error */
       if (str[strlen(str) - 1] == 0x0a)
          str[strlen(str) - 1] = 0;

       strncpy(ntp_ptr->port, str, sizeof  ntp_ptr->port);

       status = connector_callback_continue;

done:

    return status;

}
```

Step 3: Add "ntp_group_get"

```
connector_callback_status_t  ntp_group_get(connector_remote_config_t * const remote_config)
{
       connector_callback_status_t status = connector_callback_abort;
       char str[MAX_STRLEN];
       remote_group_session_t * const session_ptr = remote_config->user_context;
       ntp_config_data_t * ntp_ptr;

       ASSERT(session_ptr != NULL);
       ASSERT(session_ptr->group_context != NULL);

       ntp_ptr = session_ptr->group_context;

       switch (remote_config->element.id)
       {
```

```
            case connector_setting_ntp_server:
                ASSERT(remote_config->element.type == connector_element_type_string);
                remote_config->response.element_value->string_value = ntp_ptr->server;
                APP_DEBUG("ntp_group_get: ntp_server [%s]\n", ntp_ptr->server);
                break;
            case connector_setting_ntp_port:
                ASSERT(remote_config->element.type == connector_element_type_string);
                remote_config->response.element_value->string_value = ntp_ptr->port;
                APP_DEBUG("ntp_group_get: ntp_port [%s]\n", ntp_ptr->port);
                break;
            default:
              printf("%s: Unrecognized ntp setting\n", __FUNCTION__);
              ASSERT(0);
              break;
    }

    status = connector_callback_continue;

    return status;
}
```

Step 4: Add "ntp_group_set"

```
connector_callback_status_t  ntp_group_set(connector_remote_config_t * const remote_config)
{
     connector_callback_status_t status = connector_callback_continue;
     remote_group_session_t * const session_ptr = remote_config->user_context;
     ntp_config_data_t * ntp_ptr;

     ASSERT(session_ptr != NULL);
     ASSERT(session_ptr->group_context != NULL);
     ASSERT(remote_config->element.value != NULL);

     ntp_ptr = session_ptr->group_context;


     printf("%s: id = %d\n", __FUNCTION__, remote_config->element.id);

     switch (remote_config->element.id)
     {
        case connector_setting_ntp_server:
            ASSERT(remote_config->element.type == connector_element_type_string);
            strcpy(ntp_ptr->server, remote_config->element.value->string_value);
            break;
        case connector_setting_ntp_port:
            ASSERT(remote_config->element.type == connector_element_type_string);
            strcpy(ntp_ptr->port, remote_config->element.value->string_value);
            break;
       default:
            printf("%s: Unrecognized ntp setting\n", __FUNCTION__);
            ASSERT(0);
            break;
    }

    return status;
}
```

Step 5: Add "ntp_group_end"

```
connector_callback_status_t ntp_group_end(connector_remote_config_t * const remote_config)
{
    connector_callback_status_t status = connector_callback_abort;
    remote_group_session_t * const session_ptr = remote_config->user_context;
    ntp_config_data_t * ntp_ptr = NULL;

    ASSERT(session_ptr != NULL);
    ASSERT(session_ptr->group_context != NULL);

    ntp_ptr = session_ptr->group_context;

    printf("%s: id = %d\n", __FUNCTION__, remote_config->element.id);

    if (remote_config->action == connector_remote_action_set)
    {
        if (uci_write_config_var("ntpclient", "ntp_server",  ntp_ptr->server, "general") < 0)
        {
            goto done;
        }
```

```
        if (uci_write_config_var("ntpclient", "ntp_server",  ntp_ptr->port, "general") < 0)
        {
            goto done;
        }
    }

    status = connector_callback_continue;

done:

    if (ntp_ptr != NULL)
    {
        free(ntp_ptr);
    }

    return status;
}
```

Step 6: Add "ntp_group_cancel"

```
void ntp_group_cancel(connector_remote_config_cancel_t * const remote_config)
{
    remote_group_session_t * const session_ptr = remote_config->user_context;

    if (session_ptr != NULL)
    {

        free(session_ptr->group_context);
    }

    return;
}
```

Build and run the new image in the Wind River Workbench

Follow these instructions;

1. Open the Wind River Workbench.
2. Select Project => Build Project.
3. Select Run => Run As => Linux Application Process

Viewing the menu in Device Manager

To view the new NTP configuration menu in Device Manager the old instance of the device must be removed and re-added. Follow these instructions;

1. Log into your account on "login.etherios.com".
2. Click on the Devices menu within the Device Manager tab.
3. Select your device.
4. Click on the "Remove Devices" button from the Devices page toolbar to remove your device.
5. Click on the "Add Devices" button from the Devices page toolbar to add your device.

If you cannot properly view the new NTP configuration menu, try logging out of your Device Manager account and close the browser. Now if you open the browser and log back in you should be able to view the NTP configuration menu correctly.