# s113_nrf52 migration document

## Introduction to the s113_nrf52 migration document

### About the document

This document describes how to migrate to new versions of the s113 SoftDevices. The s113_nrf52 release notes should be read in conjunction with this document.

For each version, we have the following sections:

- "Required changes" describes the changes that need to be done in the application when migrating from an older version of the SoftDevice.
- "New functionality" describes how to use new features and functionality offered by this version of the SoftDevice. **Note:** Not all new functionality may be covered; the release notes will contain a full list of new features and functionality.

Each section describes how to migrate to a given version from the previous version. If you are migrating to the current version from the previous version, follow the instructions in that section. To migrate between versions that are more than one version apart, follow the migration steps for all intermediate versions in order.

**Example:** To migrate from version 5.0.0 to version 5.2.0, first follow the instructions to migrate to 5.1.0 from 5.0.0, then follow the instructions to migrate to 5.2.0 from 5.1.0.

# s113_nrf52_7.0.1

This section describes how to use the new features of s113_nrf52_7.0.1 when migrating from s112_nrf52_6.1.1. The s113_nrf52_7.0.1 has changed the API compared to s112_nrf52_6.1.1.

## Required changes

The application can no longer use the option `BLE_COMMON_OPT_ADV_SCHED_CFG`. The advertiser will always use improved scheduling. This was previously defined as `ADV_SCHED_CFG_IMPROVED`.

The macros `NRF_SOC_APP_PPI_CHANNELS_SD_DISABLED_MSK`, `NRF_SOC_APP_PPI_CHANNELS_SD_ENABLED_MSK`, `NRF_SOC_APP_PPI_GROUPS_SD_DISABLED_MSK`, and `NRF_SOC_APP_PPI_GROUPS_SD_ENABLED_MSK` are removed. The application can use the macros `NRF_SOC_SD_PPI_CHANNELS_SD_ENABLED_MSK` and `NRF_SOC_SD_PPI_GROUPS_SD_ENABLED_MSK` to deduce the PPI channels and groups available to the application.

## New functionality

**Connection event trigger**

When enabled, this feature will trigger a task at the start of connection events. The application can configure the SoftDevice to trigger a task every N connection events starting from a given connection event counter.

**API Updates**

- `sd_ble_gap_next_conn_evt_counter_get()`. This API can be used to retrieve the next connection event counter.
- `sd_ble_gap_conn_evt_trigger_start()`, `sd_ble_gap_conn_evt_trigger_stop()`. These APIs can be used to start and stop triggering a task on connection events.

**Usage**

The code snippet below illustrates how to configure the SoftDevice to toggle the GPIO pin 13, every second connection event, starting at connection event 10. The code snippet stops the connection event trigger when the connection parameters are updated.

```
void on_ble_evt(const ble_evt_t * p_ble_evt)
{
  if (p_ble_evt->header.evt_id == BLE_GAP_EVT_CONNECTED)
  {
    uint16_t conn_handle = p_ble_evt->evt.gap_evt.conn_handle;
```

```
    ble_gap_conn_event_trigger_t trigger_params;
    trigger_params.ppi_ch_id = 0;
    trigger_params.task_endpoint = &NRF_GPIOTE->TASKS_OUT[0];
    trigger_params.conn_evt_counter_start = 10;
    trigger_params.period_in_events = 2;

    sd_ble_gap_conn_evt_trigger_start(conn_handle, &trigger_params);
  }
  else if (p_ble_evt->header.evt_id == BLE_GAP_EVT_CONN_PARAM_UPDATE)
  {
    uint16_t conn_handle = p_ble_evt->evt.gap_evt.conn_handle;
    sd_ble_gap_conn_evt_trigger_stop(conn_handle);
  }
}

int main(void)
{
  /* Configure GPIOTE */
  NRF_GPIO->DIRSET = (1 << 13);
  NRF_GPIOTE->CONFIG[0] = (GPIOTE_CONFIG_POLARITY_Toggle << GPIOTE_CONFIG_POLARITY_Pos)
                          | (13 << GPIOTE_CONFIG_PSEL_Pos)
                          | (GPIOTE_CONFIG_MODE_Task << GPIOTE_CONFIG_MODE_Pos);

  /* Enable the BLE Stack and connect device */
  sd_ble_enable(...);
  sd_ble_gap_connect(...);

  [...]
}
```

## Configurable inclusion of Central Address Resolution (CAR) characteristic and Peripheral Preferred Connection Parameters (PPCP)

### API Updates

- `BLE_GAP_CFG_CAR_INCL_CONFIG`. This allows the application to include or exclude the CAR characteristic from the GAP Service.
- `BLE_GAP_CFG_PPCP_INCL_CONFIG`. This allows the application to include or exclude the PPCP characteristic from the GAP Service.

For the above inclusion configuration APIs, the application can use:

- `BLE_GAP_CHAR_INCL_CONFIG_INCLUDE:` The characteristic is included.
- `BLE_GAP_CHAR_INCL_CONFIG_EXCLUDE_WITH_SPACE:` The characteristic is excluded, but the SoftDevice will reserve the attribute handles which are otherwise used for this characteristic.
- `BLE_GAP_CHAR_INCL_CONFIG_EXCLUDE_WITHOUT_SPACE:` The characteristic is excluded.

### Usage

The code snippet below illustrates how to configure the SoftDevice to exclude both CAR and PPCP from the GAP Service.

```
int main(void)
{
  ble_cfg_t cfg;


  /* Exclude CAR from GAP service, but reserve the ATT Handles that will otherwise be used up by CAR. */
  cfg.gap_cfg.car_include_cfg = BLE_GAP_CHAR_INCL_CONFIG_EXCLUDE_WITH_SPACE;
  sd_ble_cfg_set(BLE_GAP_CFG_CAR_INCL_CONFIG, &cfg, ..);


  /* Exclude PPCP from GAP service, but reserve the ATT Handles that will otherwise be used up by PPCP. */
  cfg.gap_cfg.ppcp_include_cfg= BLE_GAP_CHAR_INCL_CONFIG_EXCLUDE_WITH_SPACE;
  sd_ble_cfg_set(BLE_GAP_CFG_PPCP_INCL_CONFIG, &cfg, ..);

  /* Enable the BLE Stack. */
  sd_ble_enable(...);

  [...]
}
```