G.FIT Firmware Updater

Application Note

D00001720 **Rev 0.6**

T +1 403.932.9292 F +1 403.932.6521 www.dynastream.com www.thisisant.com
#201, 100 Grande Blvd., Cochrane, Alberta, Canada T4C 0S4

Copyright Information and Usage Notice

Revision History

Revision	Effective Date	Description
0.1	March 2018	Creation of document
0.2	March 2018	Incorporated matt's comments
0.3	April 2018	Post meeting update. Added section 5.
0.4	April 2018	Now with BLE updates performed using nRF Connect, and updated CRR information.
0.5	April 2018	Amended to include Harrison's info regarding the limitations of using the G.FIT bootloader and manufacturing recommendations (section 2 and 4).
0.6	May 2018	Added table 1-1 (sections to read/ignore) and section 5.3.2 (where to find files needed for firmware updates). Incorporated Mike's comments. Updated headers and footers to automatically apply the correct version number, doc title, and doc number based on the document properties (See 'File' tab).

Table of Contents

1 Introduction 6
2 Related documents 7
3 Choosing between UART and BLE for production updates 8
4 Applying updates to unmodified G.FIT modules 9
4.1 Perform the update via UART 9
4.2 Perform the update via BLE 10
5 Customising G.FIT modules 13
5.1 Obtain a customer key 13
5.1.1 Load the secondary key 14
5.2 Develop a custom application 14
5.2.1 Customer reserve region 15
5.2.2 Include support for firmware updates 16
5.2.3 Testing recommendations 17
5.3 Generate signed update packages 17
5.3.1 Edit the configuration file 17
5.3.2 Gather the input files 18
5.3.3 Generate DFU package using dfu-util.exe 19
5.3.4 Combine OEM SD + BL and custom application packages 20
5.4 Perform a firmware update 21
5.4.1 Perform the update via UART 21
6 Reset the G.FIT module to factory defaults 23
6.1 Remove custom application and reset application version number 23
6.2 Remove secondary key 23
7 Reference: Key and signed update combinations 24
8 Heading 1 26
8.1 Heading 2 26
8.1.1 Heading 3 26
8.2 Headings use sentence style capitalization 26
8.2.1 Like this 26
8.2.2 Not Like This 26
8.3 Captions, tables, figures, and equations 26
8.4 Bullets and numbered text 27
8.5 Notes, bold text, italicized text, and document names 28
8.6 Watermarks 28
9 Second 'Heading 1' heading 29
Appendix A - Appendix H1 30
A.1 Appendix H2 30
A.1.1 Appendix H3 30

List of Figures

Figure 1-1. G.FIT memory architecture 6
Figure 5-1. G.FIT custom memory architecture 14
Figure 8-1. All captions include chapter number 27

List of Tables

Table 1-1. Which document sections to read/ignore 6
Table 3-1. Protocol selection 8
Table 4-1. Firmware update tools (UART) 9
Table 4-2. nrfutil UART parameters 10
Table 4-3. Firmware update tools (BLE) 10
Table 5-1. Available flash region 13
Table 5-2. Example starting CRR 16
Table 5-3. Example of updated CRR with data loss 16
Table 5-4. Configuration file parameters 17
Table 5-5. Summary of update packages for customized modules 21
Table 5-6. Firmware update tools (UART) 21
Table 5-7. nrfutil UART parameters 22
Table 7-1. Effect of loading different keys 24
Table 7-2. Effect of loading updates signed with different keys 25
Table 8-1. Table captions go over table 26

Equation 8-1. Deriving the scale factor 26

1 Introduction

The G.FIT module provided by Dynastream is shipped with OEM firmware preinstalled. The firmware includes a SoftDevice, a default network processor application (including the G.FIT library), memory space, and a bootloader called the G.FIT firmware updater (see Figure 1-1).

Dynastream provides firmware updates including fixes and/or improvements to the SoftDevice and network processor application. The G.FIT firmware updater enables these updates to be applied securely via *Bluetooth* ® low energy or UART. The updates are provided as encrypted packages that are decoded using a primary key that is known to the bootloader.

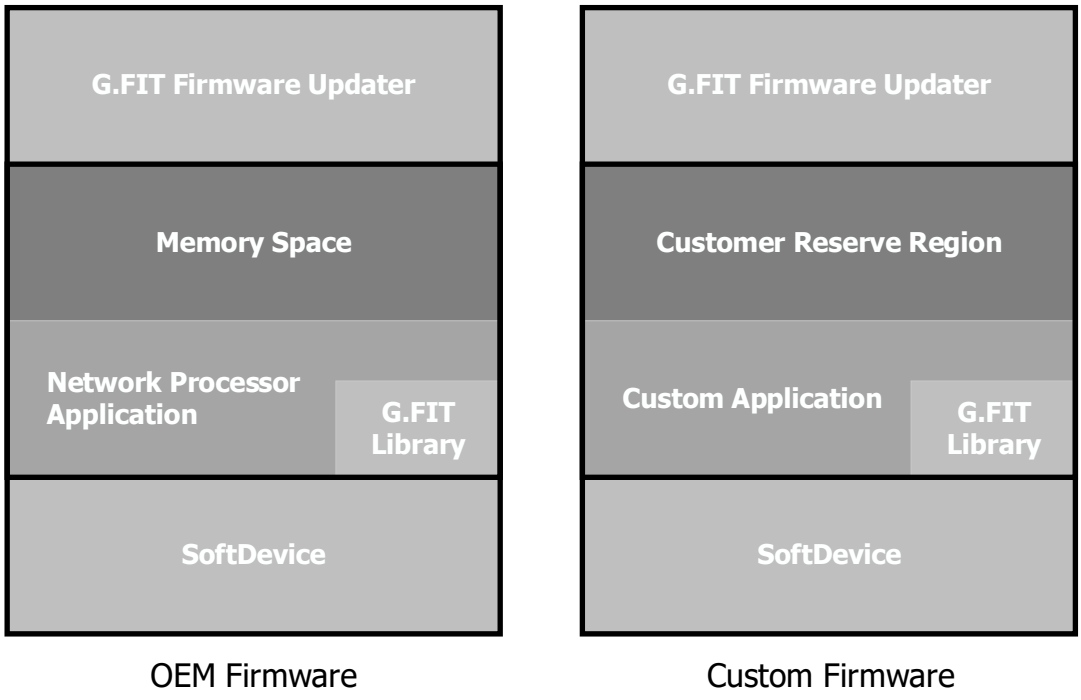


Figure 1-1. G.FIT memory architecture

The default application can also be replaced with a custom application. This process requires a secondary key, which is either a customer key generated by Dynastream (see section 5.1) or an evaluation key. Each customer key is unique to the company that requested it and should be kept secure. Performing firmware updates on modules with custom applications requires knowledge of the correct secondary key, together with correct handling of the encrypted update packages for the SoftDevice, G.FIT firmware updater, and custom application. **Failure to follow the guidance in this document can result in wiped firmware or unusable modules.**

This document describes both how to perform firmware updates for unmodified OEM modules and how to customize modules by generating and applying signed update packages. Table 1-1 indicates which document sections are relevant in each case.

Table 1-1. Which document sections to read/ignore

Goal	Section		
	1-3	4	5-7
Apply OEM updates to unmodified G.FIT module.	Read	Read	Ignore
Customize a G.FIT module and apply updates.	Read	Ignore	Read

2 Related documents

Refer to current versions of the listed documents. To ensure you are using the current versions, check www.thisisant.com or http://infocenter.nordicsemi.com/index.jsp or contact your Dynastream representative.

a. nRF Connect Bluetooth Low Energy User Guide

b. nrfutil User Guide

c. nRF5x pynrfjprog User Guide

d. G.FIT library reference documentation

e. G.FIT Fitness Equipment Modules Datasheet

f. G.FIT User Guide and Specification

g. G.FIT and Premium Module Manufacturing Considerations Application Note

3 Choosing between UART and BLE for production updates

Both UART and BLE are reasonable choices for performing device firmware updates in production, however each communication protocol has associated considerations as outlined in the table below.

Table 3-1. Protocol selection

	UART	BLE
Advantage	Significantly faster uploads than BLE (seconds vs. minutes). Most reliable method.	Does not require physical connection to the device, allowing for simpler mechanical design.
Challenge	Requires physical connection to the device.	Reliability challenges due to wireless interference, humidity, and not knowing which devices have been updated successfully. These challenges can be addressed using an isolation process, where single devices or small batches are enclosed in a Faraday cage together with the BLE host device for the duration of the update. BLE updates are typically slower than UART updates.

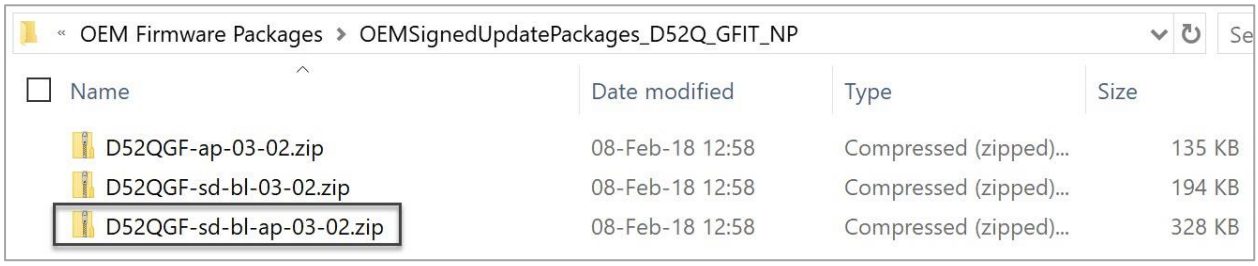
4 Applying updates to unmodified G.FIT modules

To apply firmware updates to G.FIT modules that have unmodified Dynastream firmware (SoftDevice, default network processor application, and G.FIT firmware updater), follow the guidance in this section (section 4 only). The information in later sections can be ignored.

To apply firmware updates to G.FIT modules running custom applications: skip this section and follow the guidance from section 5 onwards.

1) Download the latest GFIT_Libraries_Package.zip from thisisant.com

2) Find the firmware update package that includes the full set of OEM firmware (SoftDevice, G.FIT Firmware Updater, and G.FIT Application) under GFIT-FW-UPDATER-PKG-VERSION > OEM Firmware packages > OEMSignedUpdatePackages_D52X_GFIT_NP. The file name will be similar to: D52XGF-sd-bl-ap-XX-XX.zip. Ensure you choose the package that matches your type of G.FIT module, i.e., D25Q or D52M.



3) Follow the steps in 4.1 to perform the update via UART **or** in 4.2 to perform the update via BLE.

4.1 Perform the update via UART

Download and install nrfutil or nRF Connect and set up your hardware according to the relevant user guide (section 2). The following guidance assumes that nrfutil is used. The process for nRF Connect is similar.

Table 4-1. Firmware update tools (UART)

Tool	Intended use	Provided by
nrfutil	Command line utility used to update firmware via UART.	Nordic Semiconductor
nRF Connect	GUI version of nrfutil that supports serial communication (UART)	Nordic Semiconductor

1) Place the update package (D52XGF-sd-bl-ap-XX-XX.zip) in the same directory as nrfutil.exe.

2) Connect the G.FIT module to be updated to your computer using a USB to UART bridge connected to the appropriate pinout. The pinout varies between the Q and M module types, see datasheet for details.

3) Enter the G.FIT Firmware Updater on the module, either by using serial command 0xD8 (see *G.FIT User Guide and Specification*), or by holding the BOOT pin active low and pressing reset (refer to the datasheet for G.FIT module pin configurations).

4) Open a command line editor and enter the following:

```
nrfutil dfu serial
-ic NRF52
-p [COM PORT]
-pkg [DFU PACKAGE ZIP]
-b [BAUD RATE]
```

Where:

Table 4-2. nrfutil UART parameters

Parameter	Description	Example
[COM PORT]	Port the G.FIT is plugged into.	COM3
[DFU PACKAGE ZIP]	Filename of signed update package.	D52QGF-sd-bl-ap-03-02.zip
[BAUD RATE]	Update speed in bits per second.	115200

For example, to perform the update at 115200 bits per second:

```
nrfutil dfu serial -p COM3 -pkg D52XGF-sd-bl-ap-03-02.zip -b 115200
```

5) Press enter to run the command and perform a full DFU procedure using the serial UART connection. **This will overwrite the existing firmware.**

The G.FIT module now contains the new firmware. Query the updated G.FIT module to confirm that the new version number is returned as expected. See serial request 0xE2 0xC0 'G.FIT version number' in the *G.FIT User Guide and Specification*.

4.2 Perform the update via BLE

Download and install nRF Connect and the nRF Connect Bluetooth Low Energy app and set up your hardware according to the relevant user guide (section 2).

Table 4-3. Firmware update tools (BLE)

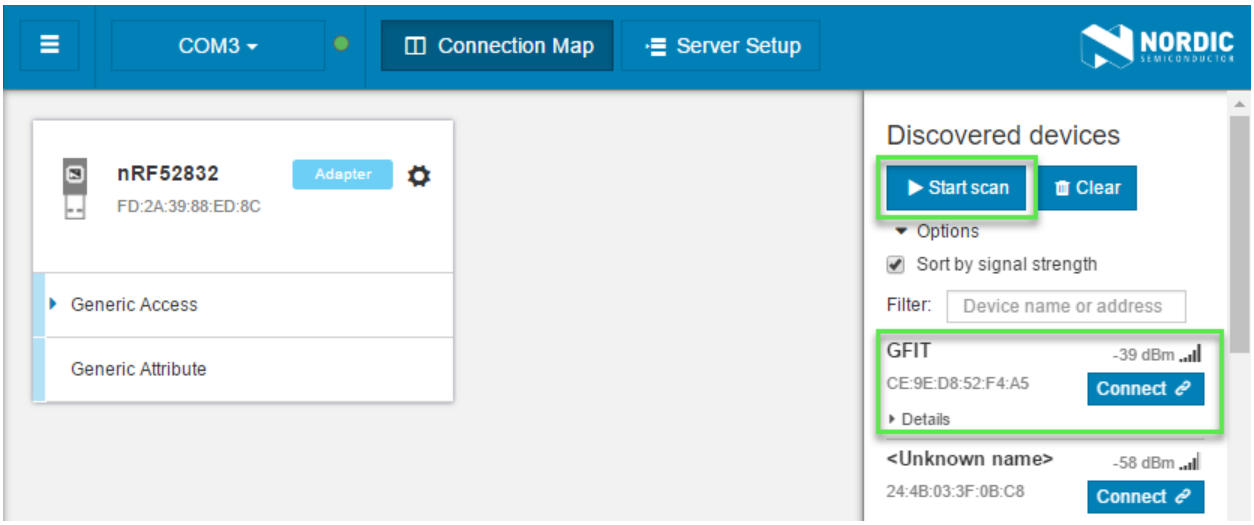
Tool	Intended use	Provided by
nRF Connect	GUI version of nrfutil that supports serial communication (UART)	Nordic Semiconductor
nRF Connect Bluetooth Low Energy	Extension app that supports communication over BLE	Nordic Semiconductor

Note: To communicate with the G.FIT module using BLE, the module must have BLE enabled (see the *G.FIT User Guide and Specification*).

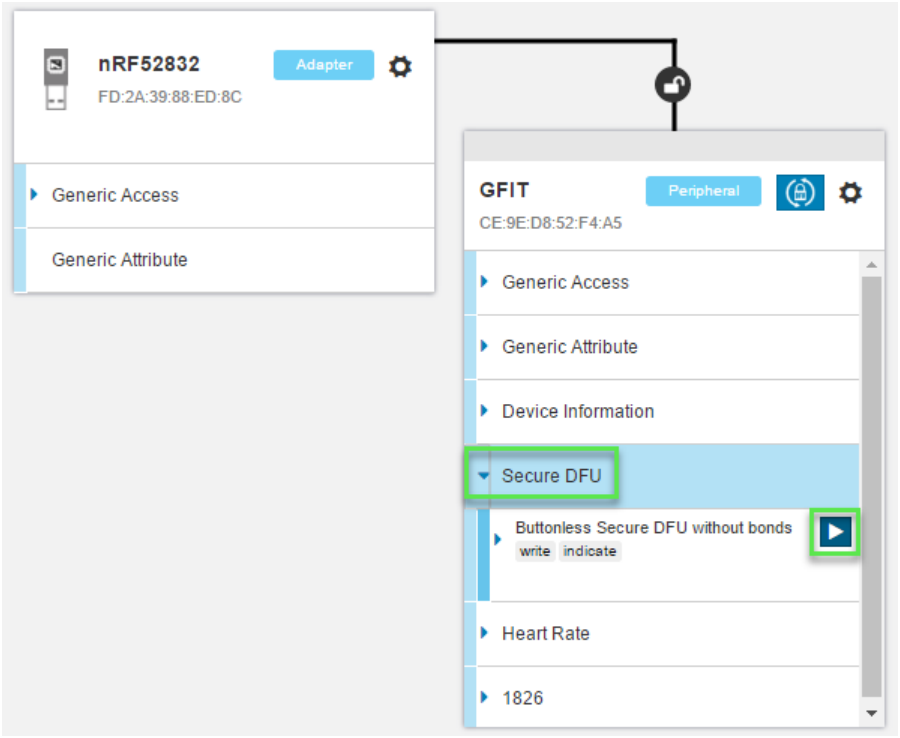
1) Open the nRF Connect Bluetooth Low Energy app

2) Connect an nRF52 Development Kit board to your computer. This board acts as the host for performing the firmware update on the target G.FIT module.

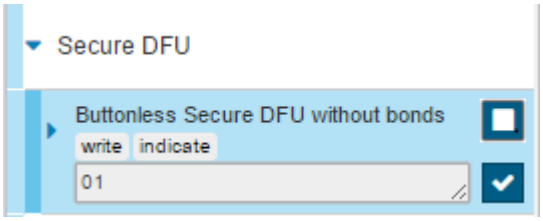
3) Click **Start scan** to find your G.FIT device, then click **Connect**.



4) In the GFIT peripheral details pane that appears, click **Secure DFU** to expand it. Then click **play** to enable notifications.



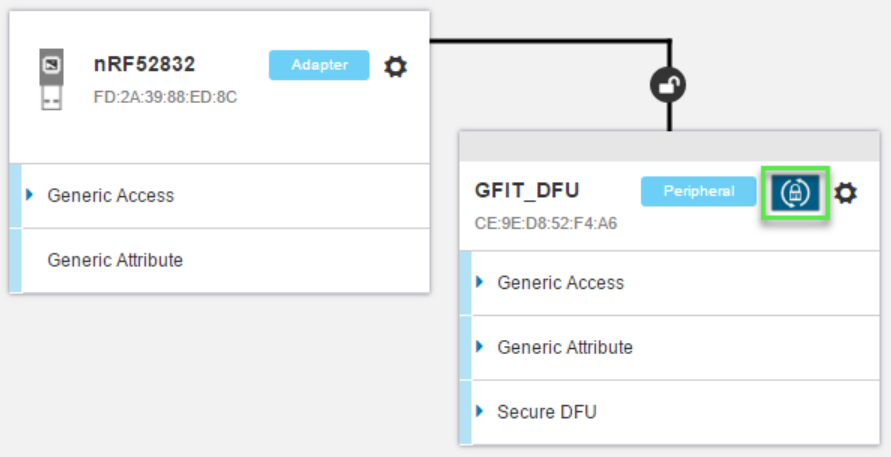
5) Click **write**, and type 01 in the text box that appears.



6) Click the checkmark to send the notification. This will start the G.FIT firmware updater in BLE mode and will cause the G.FIT module to disconnect and disappear from the screen.
Note: Sending notification '02' instead of '01' will put the device into UART transport mode.)

7) Click **Start scan** and then click **Connect** next to the device that now appears as 'GFIT_DFU'.

8) Click the **Start Secure DFU** button to open a browse window.



9) Click **Choose** and browse to the update package (D52XGF-sd-bl-ap-XX-XX.zip), then click **Start DFU** to begin the firmware update.

10) When the progress bar reaches 100%, the firmware update is complete. Click **Close**.

The G.FIT module now contains the new firmware. Observe the software version broadcast in the device information service to confirm that the G.FIT module now contains the upgraded software version.

5 Customising G.FIT modules

The nRF52 SoC used by the G.FIT module contains 512KB flash. Part of the flash is reserved for the SoftDevice and G.FIT firmware updater. This leaves the space detailed in Table 5-1 available for a custom application. **Compiling application firmware beyond the specified region is likely to cause serious errors.**

Table 5-1. Available flash region

Variable	Value
Start address	0x29000
End address	0x72000
Size	0x49000 (approx. 300KB)

Creating an application that fits in this space and makes use of the G.FIT library and G.FIT firmware updater allows for reduced development time, by taking advantage of an existing solution to handle firmware updates securely and reliably. However as with all design decisions, this choice imposes some constraints. To keep the G.FIT firmware updater the custom application must:

- Fit within the available space.
- Respect that holding the BOOT pin active low while activating the reset pin will launch the G.FIT firmware updater.

In addition, the developer must accept the design of the G.FIT firmware updater, in particular:

- The friendly name is fixed as 'GFIT' and 'GFIT_DFU', and the manufacturer is listed as Dynastream while the bootloader is running. (While the application is running, these values can be set as desired.)
- Secure encryption is used for all updates.
- Integrating the firmware updater with a customised host application requires the use of libraries provided by Nordic Semiconductor, for which Dynastream does not provide technical support.

If these constraints cannot be respected, developers can instead remove the G.FIT firmware updater and replace it with their own solution. To do this, follow the guidance in the *G.FIT and Premium Module Manufacturing Considerations Application Note*. Once the G.FIT firmware updater has been removed, there is no way to reinstall it. To keep the G.FIT firmware updater, follow the guidance in the remainder of this document.

5.1 Obtain a customer key

A customer key is needed to be able to load custom applications on the G.FIT module. There is no fee associated with this key. To obtain one, contact your Dynastream representative and complete the G.FIT commercial licensing process.

Note that an evaluation key is provided with the G.FIT library package for testing purposes. As described in the license documentation, the evaluation key is not recommended for production deployments. This key is also widely available and therefore insecure.

The customer key will be provided together with relevant software tools and documentation:

- **The G.FIT firmware updater tool (dfu-util.exe):** used to generate firmware update packages for updating the custom application and the customer reserve region.

- **Customer key package:**
    - **Change secondary key package:** used to load the new customer key onto the G.FIT module.
    - **Customer key file:** used by the G.FIT firmware updater tool to generate firmware update packages using the correct key.
- A copy of this application note, which details how to generate and combine firmware update packages.

The new secondary key must be loaded onto the G.FIT module before the module will accept update packages that are signed using the customer key.

5.1.1 Load the secondary key

To set G.FIT up to accept custom firmware packages signed using a customer key, a matched secondary key must first be loaded on the G.FIT module. To set the secondary key on the G.FIT module, use nrfutil or nRF Connect to send the update package to the module as described in section 5.4. The update package that changes the key is named load-new-[mycompany]-key.zip, for example: load-new-dynastream-key.zip. To clear a secondary key and reset the G.FIT module firmware updater to accept only factory firmware images, see section 6.2.

5.2 Develop a custom application

The G.FIT firmware updater is built with a modified Nordic SDK. Custom applications should be built using the G.FIT library provided by Dynastream. See the help files included in the library together with the *G.FIT User Guide and Specification* for details.

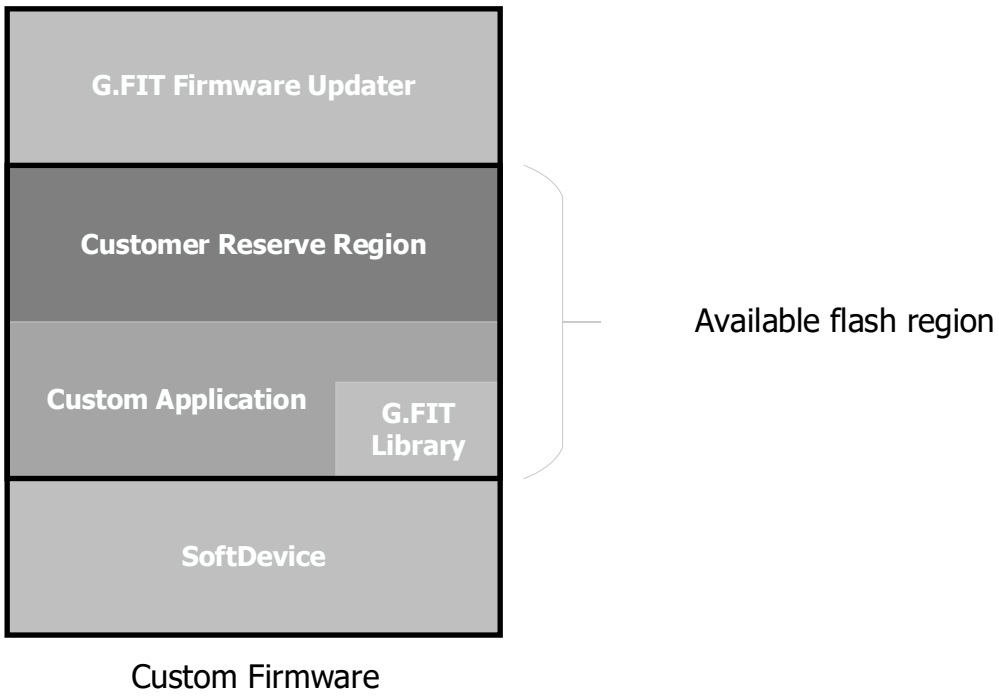


Figure 5-1. G.FIT custom memory architecture

The application may define and use a customer reserve region for data retention and must include support for launching the G.FIT firmware updater, as described in the following sections. Once defined, the custom

application and optional customer reserve region image are used to generate signed update packages (section 5.3) which are then loaded on to the module (section 5.4).

5.2.1 Customer reserve region

The customer reserve region (CRR) is a designated space that can be used as non-volatile memory by the application. The CRR is typically used to store calibration data, product information, user data and so on. Data within the CRR can be preserved during firmware updates or overwritten depending on which update package is sent to the device (see Table 5-5). To overwrite data in the CRR, define the new data in a CRR hex image and then send the CRR update package (i.e., _customer_crr_all.zip).

Possible methods of generating hex images include:

- Program desired data to a sample device and dump the intel hex record using nrfjprog --readcode.
- Build an intel hex record using binary files and a binary to hex tool, e.g.,
    - Binary to Intel Hex Converter (ARMKEIL)
    - Open source Bin2Hex (Python)
- Build an intel hex record from scratch using Python

The size of the CRR is configurable and is defined using a configuration file (config.yml – see section 5.3.1). When defining the CRR size, consider:

- Memory reserved for the CRR is no longer available for the application.
- Changes to the size of CRR **can result in loss of data** stored within the CRR. It is therefore recommended that the CRR is not changed after it is first defined.
- If no CRR is needed, the size can be set to 0.

5.2.1.1 Changing the size of the CRR

If the size of the CRR needs to be changed, then the following precautions will minimize the data loss incurred:

- Before loading the new firmware, read the existing data stored in the CRR using the nrfjprog --readcode command, and save the result as a backup.
- Ensure that the start addresses and sizes of the data values that the custom application references are adjusted according to the new size and structure. Note that space is added or removed from the start of the region. The example below illustrates this.
- If new data needs to be loaded into the CRR, then ensure that the new CRR and new custom application hex files are used to generate the new firmware update packages. When performing the firmware update, first send the update containing the new custom application, and then send the CRR update package (i.e., _customer_crr_all.zip).

Example:

This table shows a baseline CRR chosen early in product development. 5 pages are allocated to the CRR, and the application references them by start page relative to the start of the region.

Table 5-2. Example starting CRR

Page	Contents	CRR start page	CRR pages used
4	BLE Peer Manager	2	3
3	BLE Peer Manager		
2	BLE Peer Manager		
1	Product Information	1	1
0	Calibration Data	0	1

Consider increasing the size of the CRR to 8 without adjusting the start page and pages used values that the application expects. The CRR is now extended by 3 pages at the start of the region, with the intention of preserving the original data and inserting 'User Data' into the new pages. If the application tries to run in this state, it will not find the original data where it expects to (i.e., at addresses 0,1, and 2) and it is likely to overwrite the new user data with calibration data. This is likely to result in data loss and errors running the application.

Table 5-3. Example of updated CRR with data loss

Page	Contents	Actual CRR start page	Start page used by application
7	BLE Peer Manager	5	2
6	BLE Peer Manager		
5	BLE Peer Manager		
4	Product Information	4	1
3	Calibration Data	3	0
2	User Data	0	-
1	User Data		-
0	User Data		-

In the event that application is updated to look for the appropriate values at the actual start addresses, then the existing data can be preserved, and the new user data can be stored in the allocated space at the beginning of the new, larger, CRR.

5.2.2 Include support for firmware updates

To preserve the ability to install firmware updates, the custom application must be integrated with the G.FIT firmware updater. This is done by defining a way to trigger the jump to the bootloader (for example, upon a user action) and calling the library function FEP_Enter_Bootloader, shown below.

Code sample:

```
/**@brief function for preparing reset and jumping into the firmware updater
*
*@param    transport_type       Type of DFU transport to use when in the firmware
updater: BLE (0x01) or UART (0x02)
*@return   ERROR  If not successful, otherwise jump and stay in firmware updater.
**
*/
fe_module_lib_return_t FEP_Enter_Bootloader(uint32_t transport_type)
```

If a new application is installed on the G.FIT module that does not include this code, or a way to call the code, then **future firmware updates will be compromised (or impossible)**. In this case, the firmware updater mode would only be able to be entered using the reset button or pin on the G.FIT module, which could be difficult to access depending on the mechanical design of the product.

5.2.3 Testing recommendations

Custom application packages should be thoroughly tested prior to public release. Specifically, these tests should be completed on development units before release to production units:

- Test the application to ensure there are no infinite loops/bootloops.
- Use a nrfjprog --readcode command before and after the upgrade and compare the output files to confirm the original module code and new module code are different (meaning the upgrade was successful).
- Check that the CRR contents are not destroyed (particularly important when CRR changes are made).
- Test that the firmware updater still functions after the update. (Note that regression test tools are not provided by Dynastream.)

Note that the evaluation key may be used for development and testing purposes if the customer key is not yet available. This can be found in the 'Evaluation License' folder of the GFIT-FW-UPDATER-PKG available for download as part of the G.FIT libraries package.

5.3 Generate signed update packages

Signed update packages are zip files that contain the set of files needed to update a G.FIT module. The update packages are generated using the tool dfu-util.exe. The tool generates packages based on the information in the configuration file.

5.3.1 Edit the configuration file

The configuration file is written in YAML (yet another markup language) and defines the parameters listed in the following table:

Table 5-4. Configuration file parameters

Element	Description
CUSTOMER_RESERVE_REGION_SIZE	Size of non-volatile memory located at the end of the available flash region (section 5.2.1). This definition must be included, even if the size is set to 0.
KEYS - CUSTOMER	Secondary key provided to licensed developer (section 4).
AP – FILE	Filename of custom application firmware intel hex image.
AP – VERSION	Custom application firmware version number (uint32). The firmware updater will reject any versions that are less than or equal to the current version number. To reset, see section 6.
CRR – FILE	Filename of customer reserve region hex image. Delete or comment out this section if no CRR update package is required.
COMBINE – FILES	List of files to be combined (section 5.3.4).

A configuration file is provided in the secondary keys package that is similar to the sample configuration file shown below. Note that the section marked 'Do not change' must not be modified in any way. Only the parameters listed in Table 5-4 may be changed.

```
#########################################################################
```

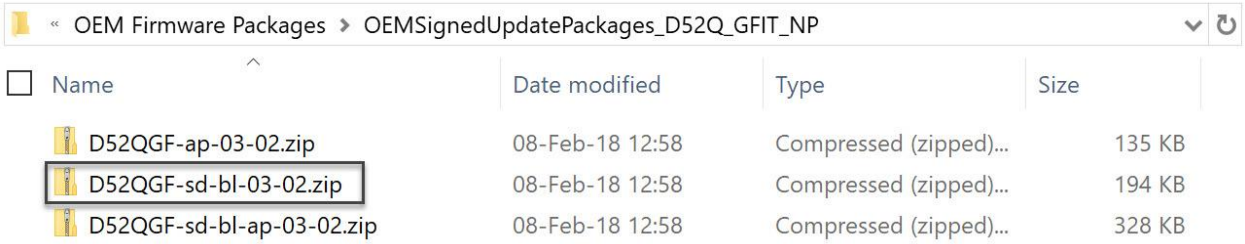
```
# DO NOT CHANGE
#
# Customers DO NOT CHANGE these values when building DFU Packages
#
HW_VERSION:     52
SD_REQ:         0x94

MEMORY:
    BL:         [0x72, 0x0B]
    SD:         [0x01, 0x28]

# Product Reserve Regions are flash pages that protected during DFU
PRODUCT_RESERVE_REGION_SIZE:    3

########################################################################

# Customer Reserve Regions are flash pages that protected during DFU
# Set this according to the custom app's size and/or storage needs
CUSTOMER_RESERVE_REGION_SIZE:   X


# Customer Keys
# Specify the name of the file that contains your custom g.fit firmware updater key
KEYS:
    CUSTOMER:   new-customer-key.keys

# Application Firmware
# Specify the name of the .hex file that contains your custom application
AP:
    FILE:       ap.hex
    VERSION:    0

# Customer Reserve Region Firmware
# Specify the .hex file that contains customer reserve region data.
# If you don't want to provide a CRR.hex file, then delete or comment out this
# section so that the DFU utility can run without errors.
CRR:
    FILE:       crr.hex

COMBINE:
FILES:
- _customer_ap.zip
- _oem_sd_bl.zip
```

5.3.2 Gather the input files

The tool, dfu-util.exe, requires these files as inputs:

- A file containing the secondary key: file.keys (see section 5.1).
- A hex image of the custom application.
- A hex image of the CRR (only needed if generating a CRR update package, see section 5.2.1).
- An edited configuration file: config.yml (see section 5.3.1).

- A copy of the OEM SoftDevice and firmware updater package (optional but needed if generating a combined update package as shown in section 5.3.4). This file is found in the GFIT_Libraries_Package.zip under GFIT-FW-UPDATER-PKG-VERSION > OEM Firmware packages > OEMSignedUpdatePackages_D52X_GFIT_NP. The file name will be similar to: D52XGF-sd-bl-XX-XX.zip. Choose the package that matches your type of G.FIT module, i.e., D25Q or D52M.



5.3.3 Generate DFU package using dfu-util.exe

To generate a firmware update package containing a custom application:

1) Open a command line editor

2) Place the configuration file, hex files, and key file into the same directory as dfu-util.exe

```
$ls
ap.hex     crr.hex       new-customer.keys     config.yml     dfu-util.exe
```

3) Run dfu-util.exe with the configuration file as the only parameter

```
$ ./dfu-util.exe config.yml
**************************************
Please select:
0: Build All
1: Build DFU Package Customer AP
2: Build DFU Package Customer Reserve Region
3: Combine SD+BL and AP .ZIP DFU Packages
To exit press Ctrl-C
```

4) Type 0, 1, or 2 and press ENTER.

This results in the output shown below. The generated zip files will appear in the current directory. **Do not make any modifications** to the firmware update packages generated by the tool.

5.3.3.1 0: Build all

```
0
*** Building Customer AP DFU package
Generating: _customer_ap.zip
*** Building Customer Reserve Region DFU packages
Generating: _customer_crr_all.zip
Done
```

5.3.3.2 1: Build DFU Package Customer AP

```
1
*** Building Customer AP DFU package
Generating: _customer_ap.zip
Done
```

5.3.3.3 2: Build DFU Package Customer Reserve Region

```
2
*** Building Customer Reserve Region DFU packages
Generating: _customer_crr_all.zip
Done
$ls
_customer_ap.zip  config.customer.yml dfu-util.log   _customer_crr_all.zip
crr.hex new-customer.keys      ap.hex dfu-util.exe
```

5.3.4 Combine OEM SD + BL and custom application packages

When updates to the SoftDevice and/or G.FIT firmware updater are made available by Dynastream, these must be combined with the custom application package before being sent to the G.FIT module. Updating the G.FIT module with the uncombined packages may wipe the custom application (see section 6).

To generate a combined firmware update package:

1) Edit the new configuration file to reflect the desired values (see section 5.3.1).

2) Save the file.

3) Open a command line editor

4) Place the new configuration file, custom application zip file, OEM package zip file, and the key file into the same directory as dfu-util.exe

```
$ls
customer_ap.zip   oem_sd_bl.zip new-customer.keys    config.yml    dfu-util.exe
```

5) Run dfu-util.exe with the configuration file as the only parameter

```
$ ./dfu-util.exe config.yml
************************************
Please select:
0: Build All
1: Build DFU Package Customer AP
2: Build DFU Package Customer Reserve Region
3: Combine SD+BL and AP .ZIP DFU Packages
To exit press Ctrl-C
```

6) Type 3 and press ENTER.

This results in the output shown below.

```
3
*** Combine SD+BL and AP DFU packages
Done
```

The generated zip file (_combined_sd.bl.ap.zip) will appear in the current directory. **Do not make any modifications** to the firmware update packages generated by the tool.

5.4 Perform a firmware update

The same process is used to load firmware updates and to change the secondary key. Choose the appropriate package as indicated in the table below.

Table 5-5. Summary of update packages for customized modules

Update package	Usage
load-new-mycompany-key.zip	Use to load the secondary key (see section 5.1.1). This must be done before any other packages that use this key will be accepted by the device. Once loaded, the key is remembered.
_customer_ap.zip	Use to update the custom application without changing the SoftDevice or G.FIT firmware updater. Data within the CRR as defined in config.yml is not overwritten by loading this package.
_customer_crr_all.zip	Use to update the CRR. This overwrites the entire CRR without changing the custom application, SoftDevice or G.FIT firmware updater.
oem_sd_bl.zip	**Do not load this package onto the module.** Use it only to create the _combined_sd_bl_ap.zip package (see section 5.3.4).
_combined_sd_bl_ap.zip	Use to update the custom application and/or the SoftDevice and G.FIT firmware updater. Data within the CRR as defined in config.yml is not overwritten by loading this package.

Follow the steps in section 5.4.1 to perform the update via UART. **Or** follow the steps in section 4.2 to perform the update via BLE but using a signed update package from Table 5-5.

5.4.1 Perform the update via UART

Download and install nrfutil or nRF Connect and set up your hardware according to the relevant user guide (section 2). The following guidance assumes that nrfutil is used. The process for nRF Connect is similar.

Table 5-6. Firmware update tools (UART)

Tool	Intended use	Provided by
nrfutil	Command line utility used to update firmware via UART.	Nordic Semiconductor
nRF Connect	GUI version of nrfutil that supports serial communication (UART)	Nordic Semiconductor

1) Place the update package (chosen from Table 5-5) in the same directory as nrfutil.exe.

2) Connect the G.FIT module to be updated to your computer using a USB to UART bridge connected to the appropriate pinout. The pinout varies between the Q and M module types, see datasheet for details.

3) Enter the G.FIT firmware updater on the module, using the appropriate method based on your module.

   a) To run the G.FIT firmware updater from G.FIT modules running the default firmware, either:

   - Use serial command 0xD8 (see *G.FIT User Guide and Specification*), or,
   - Hold the BOOT pin active low and press reset (refer to the datasheet for G.FIT module pin configurations).

b) To run the G.FIT firmware updater from G.FIT modules running a custom application: take the action defined by the custom application (see section 5.2.25.2.2).

4) Open a command line editor and enter the following:

```
nrfutil dfu serial
-ic NRF52
-p [COM PORT]
-pkg [DFU PACKAGE ZIP]
-b [BAUD RATE]
```

Where:

Table 5-7. nrfutil UART parameters

Parameter	Description	Example
[COM PORT]	Port the G.FIT is plugged into.	COM3
[DFU PACKAGE ZIP]	Filename of signed update package.	_combined_sd_bl_ap.zip
[BAUD RATE]	Update speed in bits per second.	115200

For example, to perform the update at 115200 bits per second:

```
nrfutil dfu serial -p COM3 -pkg _combined_sd_bl_ap.zip -b 115200
```

5) Press enter to run the command and perform a full DFU procedure using the serial UART connection. **This will overwrite the existing firmware.**

The G.FIT module now contains the new firmware. Query the updated G.FIT module to confirm that the new version number is returned as expected.

6 Reset the G.FIT module to factory defaults

If the module needs to be reset, either to clear the custom application, or reset the secondary key, then see the relevant section below:

6.1 Remove custom application and reset application version number

To reset the custom application version number, load a firmware update package containing the OEM firmware including the default application firmware. This will wipe any data stored in the CRR.

1) First take a backup copy of the CRR data. (I.e., read the existing data stored in the CRR using the nrfjprog --readcode command, and save the result.)

2) Follow the steps in section 4. It is important to choose the full firmware update package (i.e., the zip file with '…sd-bl-ap…' in the filename) in order to reset the module.

3) The G.FIT modules can then be updated again with the desired custom application and CRR packages. The version number in the configuration file can be reset to 0 before generating the new firmware update packages.

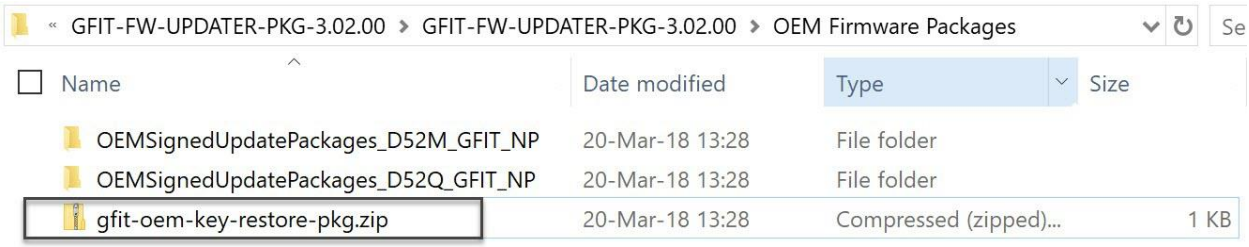
Note that this process **does not** remove the secondary key.

6.2 Remove secondary key

If removal of the secondary key is needed, then:

1) Download the latest GFIT_Libraries_Package.zip from thisisant.com

2) Find the key restoration package under GFIT-FW-UPDATER-PKG-VERSION > OEM Firmware packages > gfit-oem-key-restore-pkg.zip and then apply this update as described in section 5.4.



7 Reference: Key and signed update combinations

The following tables summarize which keys and update packages will be rejected or accepted by the G.FIT firmware updater, and what effect these updates have on the module.

Key packages:

Table 7-1. Effect of loading different keys

Key package sent	G.FIT module receiving the package			
	Unmodified module	**Module with evaluation key**	**Module with customer key**	**Module with different customer key**
load-new-mycompany-key.zip	**Accepted** G.FIT module becomes 'module with customer key'	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.
gfit-oem-key-restore-pkg.zip	Rejected G.FIT module is unchanged.	**Accepted** G.FIT module becomes 'unmodified module'.	**Accepted** G.FIT module becomes 'unmodified module'.	**Accepted** G.FIT module becomes 'unmodified module'.
load-evaluation-key.zip	**Accepted** G.FIT module becomes 'module with evaluation key'	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.
load-new-mycompany2-key.zip	**Accepted** G.FIT module becomes 'module with different customer key'	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.

Note that changing which key is loaded on the module only changes which signed update packages the module will accept. It does not change the firmware loaded on the module. To fully return a customized module to its unmodified state see section 6.

Signed update packages:

Table 7-2. Effect of loading updates signed with different keys

Signed update package sent	G.FIT module receiving the package			
	Unmodified module	**Module with evaluation key**	**Module with customer key**	**Module with customer2 key**
OEM signed update	**Accepted** Firmware is overwritten. Key is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.
Evaluation key signed update	Rejected G.FIT module is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.
Customer key signed update	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.	Rejected G.FIT module is unchanged.
Customer2 key signed update	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	Rejected G.FIT module is unchanged.	**Accepted** Firmware is overwritten. Key is unchanged.

Note that updates are only accepted when all version numbers (SoftDevice, application, and G.FIT firmware updater) in the update package are greater than or equal to the existing version numbers on the G.FIT module. The table above assumes that this is the case.

8 Heading 1

8.1 Heading 2

8.1.1 Heading 3

8.1.1.1 Heading 4

8.1.1.1.1 Heading 5

8.2 Headings use sentence style capitalization

8.2.1 Like this

8.2.2 Not Like This

Text style is 'ANT Normal'.

8.3 Captions, tables, figures, and equations

Captions are now correctly formatted by default and should automatically use the 'Caption' style (Tahoma, 8.5pt, bold, centred, line spacing single, before 0pt, after 10pt). Use sentence style capitalization, and no period at the end.

Inserting a new table should automatically give you the correct shading and borders. Modifying tables by deleting or adding rows should also preserve the correct striping. To get the correct font formatting within the table, apply the 'ANT Table Header', 'ANT Table Text Left' and 'ANT Table Text Center' styles. If a table does not have the correct shading, apply the table style 'Striped Table'.

Table 8-1. Table captions go over table

Header	Header text	ANT Table text header
Table text left	Table text centre	Table text left Text should automatically center itself vertically.
		Autofit table to window – unless for very small tables

Equations are images (layout-> 'inline' and centre by applying the 'Caption' style to the equation). Equation captions go under the equation.

$$ScaleFactor = \frac{KnownDistance}{MeasuredDistance}$$

Equation 8-1. Deriving the scale factor

Images use 'inline' layout. Centre them by applying the 'Caption' style to the image. Captions go under the image. Note that images used to be 'top and bottom' format, but this caused a lot of confusion when editing as images can jump around and get lost depending on where their anchor is positioned on the page. Using inline formatting should remove this headache.

Figure 8-1. All captions include chapter number

8.4 Bullets and numbered text

Bulleted text looks like this:

- Item one using style ANT Bullets 1
- Item two
    - Level 2 bullets using style ANT Bullets 2

Itemised lists look like this:

a. Item 1 using style ANT List 1

b. Item 2

i. Level 2 list using style ANT List 2

ii. So pretty

When you have a procedure you're writing instructions for, do this:

6) Use this style for instruction steps. The style is called ANT Instructions one and is based on the custom multi-level list 'Instruction steps'. As with all ANT styles you can find it in the style pane; or copy & paste from here.

   Indented text ANT Indent 1, to match ANT Instructions 1. Use this if you need another paragraph for the same step in the instructions. The next paragraph will automatically be ANT Instructions 1, which you can change if needed.

7) Right click the number and select 'Restart at 1' if the list starts at a higher number.

   a) Instruction sub-steps look like this.

      Indented text ANT Indent 2, to match ANT Instructions 2. Use this if you need another paragraph for the same sub-step in the instructions. The next paragraph will automatically be ANT Instructions 2, which you can change if needed.

If you need to include some sample code:

```
## Use style 'ANT Code Sample'
Lgrjalæjr
Sglsg
Hello World!
```

8.5 Notes, bold text, italicized text, and document names

Formatting can be used to communicate extra information about what you're saying. For example, **really important information that you don't want readers to miss** should be styled with 'Strong'.

Note: The 'Strong' style is also used to help people notice your notes 😉. Apply it to 'Note:' at the start of the line.

Don't italicize anything except for document names. For those use style 'Document Name', e.g., see the *ANT Message Protocol and Usage* document. Don't bother italicizing the docs listed in the related documents section.

8.6 Watermarks

To modify 'Preliminary – subject to change' watermarks, double click the header or footer area, then select the image and modify as desired.

9 Second ‘Heading 1’ heading

Each top-level heading will automatically start on a new page. No need to add page breaks.

Appendices use the styles ‘Appendix H1,2,3, or 4’ as the heading levels.

Appendix A - Appendix H1

Blah blah blah

A.1 Appendix H2

Blah blah blah

A.1.1 Appendix H3

Blah

A.1.1.1 Appendix H4

glksg