Loadable Code Settings

After the code compilation and producing the .bin file the developer needs to send the binary code to GrdUtil.exe for it to be processed and written into the dongle. The **Loadable Code Settings** dialog box serves this purpose.

This dialog box permits the:

- Import of the previously compiled code from the **.bin** file
- Conversion of the imported code into GCEXE format suitable for writing into a Guardant Code / Code Time dongle
- Writing code in GCEXE format into the Flash memory of the dongle or exporting it into an external file

Further, this dialog box provides additional services that simplify working with the loadable code.

Importing loadable code from a .bin file

Clicking the **Import code from a .bin file** button located in the top right corner of the **Loadable code settings** page launches a dialog box for selecting the . bin file from the required project.

Upon importing, GrdUtil.exe reads from the **project_name.bmap** file settings describing the dongle's memory occupied by the loadable code. After importing, the following appears in the dialog box:

Status indicator	Purpose
RAM in use	Indicates the dongle's RAM allocated for the loadable code, its start and end address
Flash in use	Indicates the dongle's Flash memory allocated for the loadable code, its start and end address as well as the number of the item storing the descriptor of the loadable code

Free memory is indicated by a green color, whereas the allocated memory – with blue. Addresses are displayed in hexadecimal format.

Binary code conversion into GCEXE format

Considering the confidentiality of the loadable code, it must not be transmitted "outside" in an open format.

Therefore, an effective routine for preparing code for writing into the dongle and the safe transfer of loadable code updates to end users was implemented in GrdUtil.exe.

GrdUtil.exe automatically converts the binary code into the .GCEXE file format containing:

- AES-encrypted source code
- AES session key (used previously for encrypting the source code) encrypted with the ***ECC160 No.1*** *public key*
- The .GCEXE file digital signature generated using the ***ECC160 No.2*** *private key*

Whereas the counterpart of the ECC keys pairs, used during the binary code conversion, is stored in the loadable code descriptor (prot. item):

- The ***ECC160 No.1****private key* for encryption
- The ***ECC160 No.2*** *public key* for the digital signature

This allows the dongle to successfully check, decrypt and execute loadable code upon addressing it.

Important information

Conversion of the code into GCEXE format is done with GrdUtil.exe utility automatically upon writing a mask into the dongle (or upon clicking the **Export GCEXE** button). This does not require any actions on the part of the developers for its execution, besides the setting of ECC160 key pair.

Loadable Code Encryption Settings

Clicking the **Loadable code encryption settings** button launches a dialog box for working with key pairs.

The dialog box is designed for generating, importing and exporting ECC160 key pairs, which are used during the conversion of binary code into GCEXE format (see previous item).

The private and open **ECC160 No.2** keys used for verifying the digital signature of encrypted code appear in the top of the dialog box (on the left and on the right, respectively).

ECC160 No.1 *key pair* for encrypting the binary code appears in the bottom of the dialog box (private key – to the left, public – to the right).

Additionally, the dialog box is provided with buttons that permit the generation of new key pairs, exporting them into external files for use in applications and importing key pair files from other projects.

Exporting the .GCEXE file. Code Update by the End User

The export of .GCEXE into an external file may be required in the event a developer needs to update the loadable code into the dongle at the end user's location.

In this case a developer needs to follow the routine described below:

1. A mechanism for updating the loadable code from within the application should be provided for at that stage of the application development. Such mechanism is implemented using the Guardant API **GrdCodeLoad** function (see ***GrdAPI.chm*** for details).
2. Upon making all required changes, the new version of the loadable code is compiled into a binary file to be **imported** into **GrdUtil.exe.**
3. To ensure proper operation of the loadable code in the remote dongle, the same key pairs, which were used for programming the dongle initially, should be used. See **Loadable code encryption settings**.
4. Clicking the **[Export GCEXE]** button generates and exports the .GCEXE into an external file.
5. If it is necessary to make the code update dongledependent (for instance, for paid updates), then the decimal ID of the remote dongle must be indicated in the dialog box, which will appear upon clicking the **[Export GCEXE]** button.
6. The loadable code saved in GCEXE format is send to the end user, who updates the contents of the dongle using the method provided for by the developer in step 1.

****

Writing the Loadable Code into a Dongle

After adjusting the loadable code settings, you need to close the dialog box and execute the **Dongle | Write into dongle** command. This will generate and write the following into the dongle:

1. A dump, containing the loadable code descriptor (among other fields) – into EEPROM memory.
2. Loadable code in GCEXE format – into Flash memory of the dongle.