ViVOconfig User Manual

Rev D
6 August 2020

ID TECH
10721 Walker Street, Cypress, CA 90630-4720
Tel: (714) 761-6368 Fax (714) 761-8880
www.idtechproducts.com

Revision History

Date	Rev	Changes Made	By
11/22/2019	A	Initial Release.	CB
05/05/2020	B	Major update to section 4: Using ViVOconfig for release of v1.0.0 and app UI changes.	CB
06/04/2020	C	Updated screenshots and documentation for most recent release.	CB
07/31/2020	D	Updated screenshots and documentation for most recent release. Major update to reference documentation for JSON options.	CB

Table of Contents

1. Introduction

ViVOconfig provides a software solution for managing standard and customized configurations across multiple ID TECH products.

1.1. Design Goals

ViVOconfig translates the various commands from multiple products into a single standard configuration file portable across product lines. A configuration file created from one product works with another product. Configurations that do not apply are ignored.

1.2. Everything Works Together

The configuration files generated by ViVOconfig are also used in ViVOstate® Terminal Management System. A user can upload the ViVOconfig configuration file used during device integration testing to ViVOstate® TMS and then push it to the appropriate devices. Conversely, when ViVOstate® reads configurations from remote devices, it saves those configurations in the ViVOconfig file format. The configuration file is portable.

1.3. Backing Up Configurations

Just like any other data file, it is strongly recommended that users create backups and store them on external media, such as a USB Thumb Drive. Backups are critical when doing development work and testing various configurations.

1.4. Cautions

1. Always test ViVOpay devices after they have been configured.
2. Do not apply any configuration setting changes via other tools or custom written software. These "back door" configuration updates will cause your ViVOconfig configuration files to be out of sync with the configuration actually programmed in the device.

1.5. Support

To obtain technical support for ViVOconfig issues, visit the ID TECH Knowledge Base or submit a question to support@idtechproducts.com (sending an email to this address automatically generates a customer support ticket). If possible, include the model and serial numbers for the device being configured.

2. Prerequisites

1. A Windows computer running Windows 7 or later.
2. Microsoft .NET Framework version 4.6.1 or higher.
3. An available USB Port
4. An ID TECH Payment Terminal with USB port. This guide uses the Augusta in examples.

3. Installation

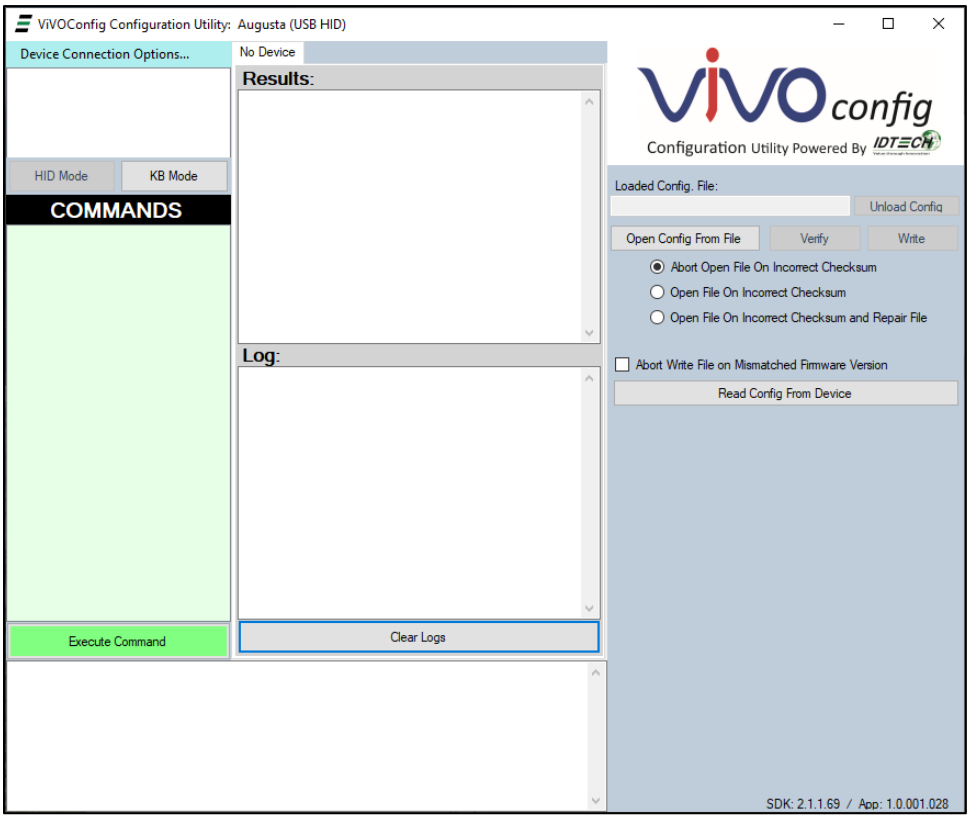
To download ViVOconfig, visit the ViVOconfig product page on the ID TECH Knowledge Base and download **ViVOconfig.zip**. Follow the steps below after downloading the ZIP file:

1. Open the ZIP file and launch **ViVOConfig_Setup.msi**.
2. Follow the Setup Wizard prompts to complete the setup.
    a. The Setup Wizard may also prompt some users to update Microsoft .NET Framework, depending on their current version.

Completing setup adds a ViVOconfig Desktop Shortcut and an ID TECH folder in the Programs List containing the ViVOconfig application.

4. Using ViVOconfig

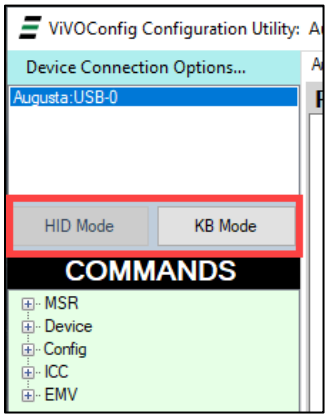
Starting ViVOconfig displays the following dialog (assuming no ID TECH device is connected to the computer):



Notice that the buttons are grayed out because no device is connected to ViVOconfig.

4.1. Connecting a Device

Next, plug the Augusta into the USB port. After a few moments, ViVOconfig autodetects the Augusta and shows its current USB Mode (HID or KB).

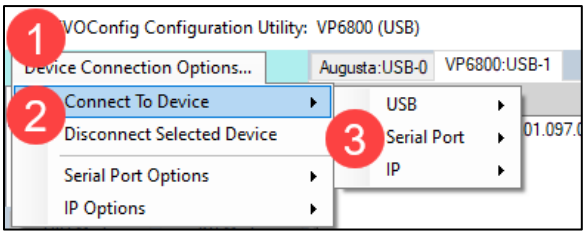


4.1.1. Switching a Device Between HID and KB Mode

ViVOconfig can easily switch between HID and KB mode for devices with that capability. Click **HID Mode** or **KB Mode** (highlighted above) as desired.

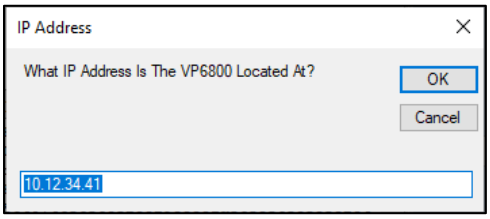
4.1.2. Additional Device Connection Options

ViVOconfig can also connect to readers that support Serial and IP connections. To connect by one of these options:



1. Select **Device Connection Options** in the upper-left corner.
2. Select **Connect to Device**.
3. Select the desired connection method.

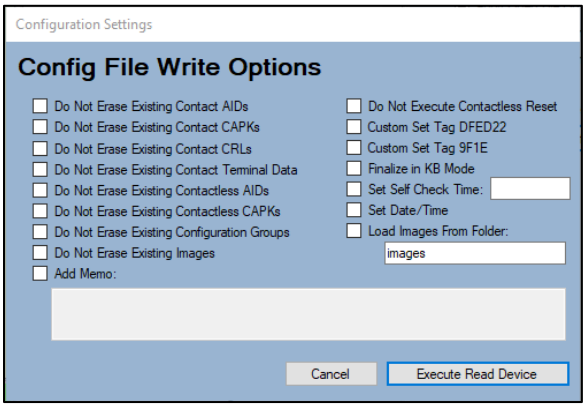
If needed, enter connection options, such as the reader's IP address.



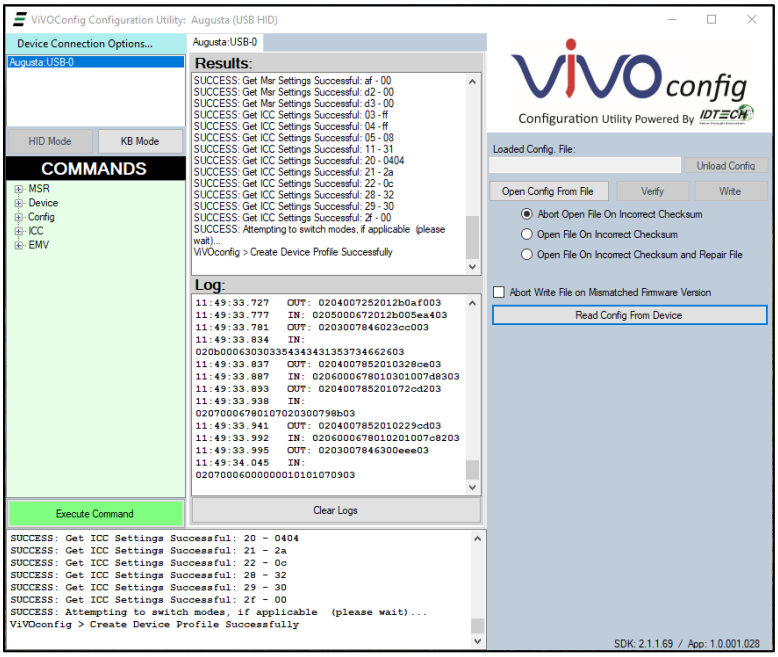
4.2. Reading the Device Configuration

To read the configuration after ViVOconfig connects to the Augusta:

1. Click the **Read Config From Device** button.
2. If desired, select **Config File Write** options in the dialog that opens.
3. Click **Execute Read Device**.



The **Notifications** window displays messages as ViVOconfig interacts with the Augusta to read its configuration:



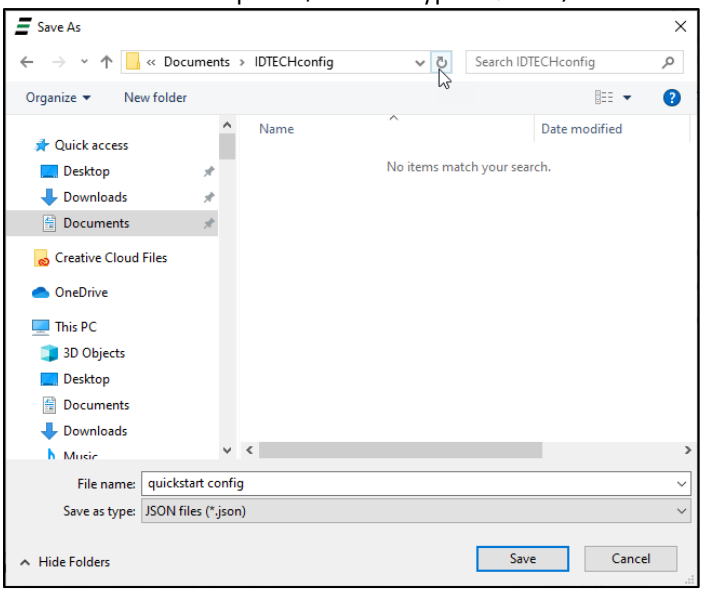
A "ViVOconfig > Create Device Profile Successfully" message indicates that ViVOconfig has finished reading from the device.

4.3. Saving the Device Configuration

After ViVOconfig reads the configuration from the Augusta, it can save that configuration to a file.

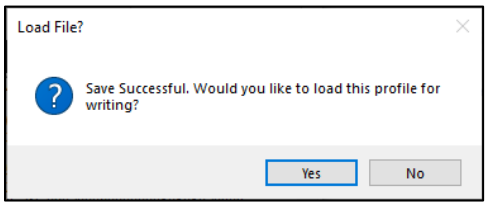
To save a configuration:

1. Read a configuration from the device as described above in **Reading the Device's Configuration**.
2. Upon a successful read, ViVOconfig automatically opens a
3. Select the desired location and enter a desired filename in the **Windows Save As** dialog (the extension is not required; the file type is JSON).



4. Click **Save**.

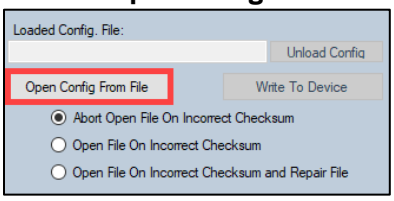
The **Results** and **Activity** windows display a message that the file has been saved. Optionally, you can load the profile for writing:



4.4. Configuring a Device with a Previously Saved Configuration

To configure a device with a previously saved configuration:

1. Connect the device to the computer and confirm that ViVOconfig recognizes it (see Initial Setup).
2. Click the **Open Config From File** button:

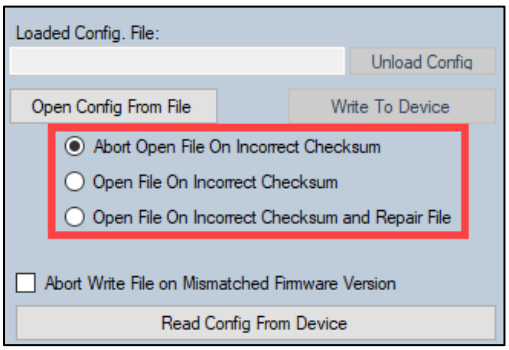


3. The File Explorer dialog shows a list of ViVOconfig JSON files. Select the desired file and click **Open**.

The **Notifications** window in the lower-left corner displays a message indicating that the application opened the file and can now write the configuration to the Augusta. The **Loaded Config. File** field displays the directory path for the currently-loaded configuration file if the configuration file passed all optional parameters and was successfully parsed.

4.4.1. Options for Opening Configuration Files

ViVOconfig provides the following options when opening configuration files, located below the **Open Config From File** button:



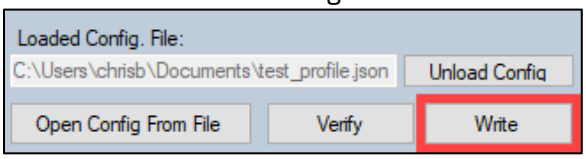
- **Abort Open File on Missing Checksum**: Opening is aborted if the file is missing a "hash" tag.
- **Open File on Incorrect Checksum:** If the "hash" tag is found and incorrect, ViVOconfig ignores it and attempts to load the file anyway.
- **Open File on Incorrect Checksum and Repair File:** If the "hash" tag is found and incorrect, ViVOconfig updates the JSON file with the correct value and attempts to load the file.

Note: All previously-created JSON files' hash values will **FAIL** in version 1.0.0 or later. Although ViVOconfig *can* load files created prior to v1.0.0, files without a valid checksum may not be safe to load. **When using options that ignore checksum errors, make sure to only load files that nobody has altered.**

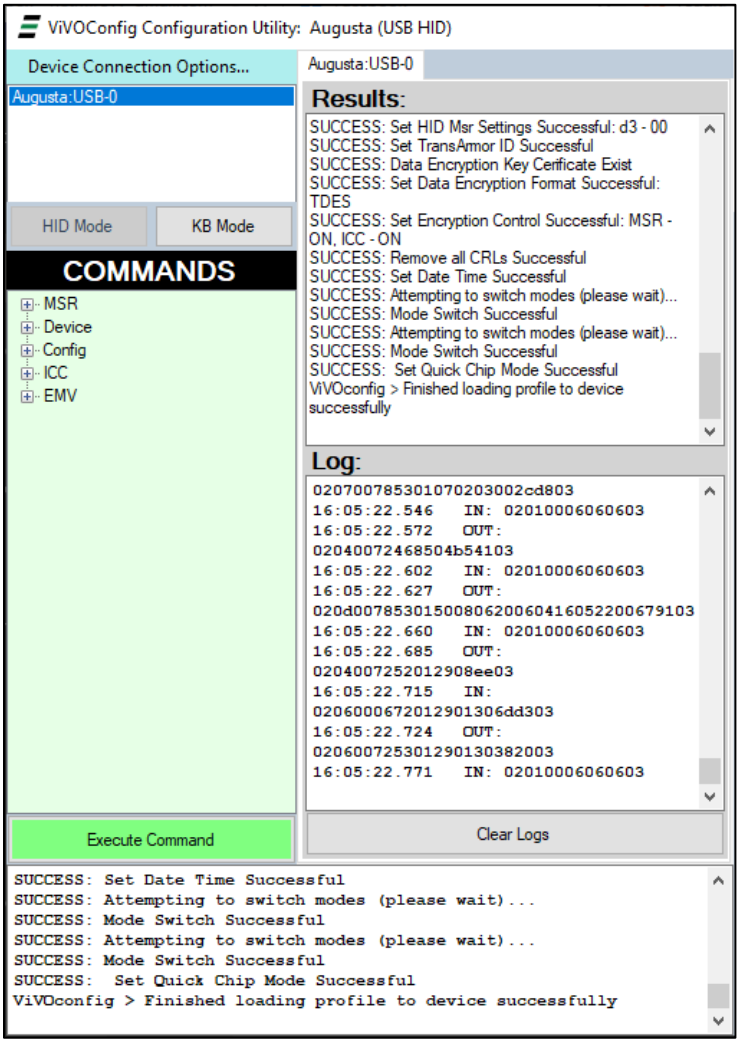
4.5. Updating the Device with the Configuration File

To update the device with a configuration file:

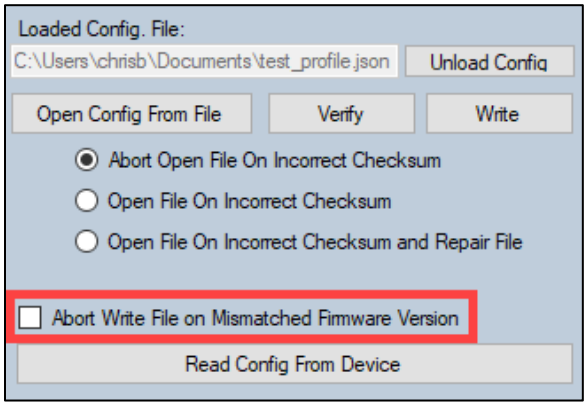
1. Load the configuration file in ViVOconfig as described above.
2. To write the selected configuration to the reader, click **Write**.



3. When prompted, click **Yes**.
4. ViVOconfig writes the configuration to the device:

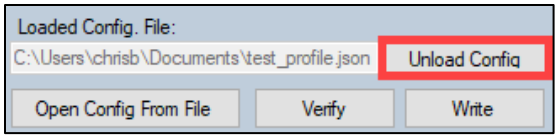


ViVOconfig provides the option to abort the update procedure if the file and device have mismatched firmware:



4.6. Unloading a Configuration File

ViVOconfig saves all settings (checkboxes, file load and save locations, the last profile loaded) and restores them on startup. Users may not want the application to attempt to load the most recently used configuration; use the **Unload Config** button to clear the currently-loaded configuration file.

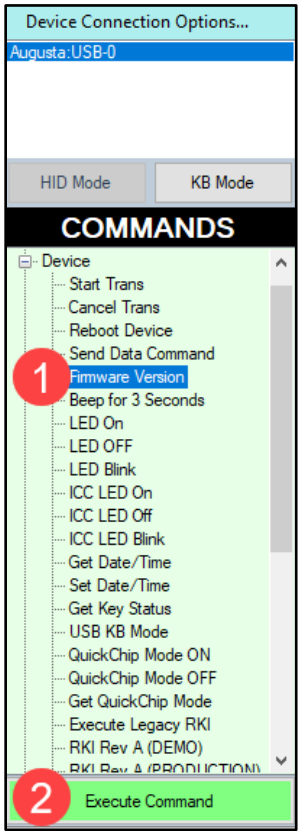


4.7. Sending Commands to a Device

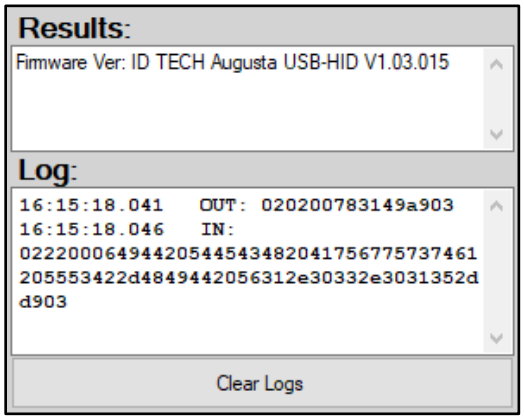
ViVOconfig can send commands to devices via the **Commands** tree. Users should only use this feature with guidance from ID TECH representatives.

Follow the steps below to send a command to the connected device:

1. Select a desired command from the **Commands** tree and click **Execute Command**.



The **Results** window displays the command response (in this case, the Get Firmware Version command). The **Data Log** window displays the command data in hexadecimal.



5. Advanced Topics

The topics below are intended for advanced users. Proceed with caution.

1. Users can directly edit JSON configuration files with any text editor (such as Notepad). Be mindful of the following:
    a. Preserve the proper JSON format and tag structure.
    b. Maintain consistency with the JSON tag names per the ViVOconfig file format specification.
    c. There are many online tools available for syntax checking and formatting that may be of assistance.
2. Because ViVOconfig reads the entire configuration of a device, there may be situations where users want to update only subset of settings, such as AIDs, CAPKs, and other specific items. Edit out all of the other configuration settings so that you have a way to only update the settings that you need.

6. Updates

ViVOconfig is a rapidly evolving product. Check the ID TECH Knowledge Base for updates.

7. Reference Documentation

This following ViVOconfig documentation describes a generic configuration interface for terminals. Not all settings and parameters are available on all terminals, nor are all setting values available on all terminals. Some settings may be placeholders for future capabilities. Whenever possible, the terminal ignores (fail safe) invalid terminal settings (parameters and values). This terminal documentation and the associated terminal firmware version are the definitive sources for determining terminal capabilities.

Unless otherwise stated, all configuration parameters are read/write. Attempts to write a read-only parameter will be ignored and the terminal parser should continue processing subsequent commands.

7.1. ID TECH JSON Conventions

This JSON schema uses **snake_case** (underscore-based) naming for property names. In other words:

- Use underscores instead of hyphens or spaces.
- Do not use CamelCase or UPPERCASE.
- Property names should not contain spaces (even though JSON can accommodate that).

7.2. TLV Conventions

When data is natively encoded using BER-TLV, TLVs are represented as a TLV blob (that is, a block of TLVs) consisting of any number of TLVs run together.

For example, terminal settings on some terminals can be represented as a block of TLVs:

```
5f3601029f1a0208409f3501219f33036028c89f4005f000f0a0019f1e085465726d69
6e616c9f150212349f160f303030303030303030303030303030309f1c08383736353433
32319f4e223130373231205761616c6b65722053742e20437970726573732c204341202c
5553412edf260101df1008656e667265737a68df110100df270100dfee150101dfee16
0100dfee170107dfee180180dfee1e08d0dc20d0c41e1400dfee1f0180dfee1b083030
303135313030dfee20013cdfee21010adfee2203323c3c
```

The block must be able to be parsed by any standard BER-TLV parser.

7.3. AID Settings

The fields below configure **AID** settings.

Array of AIDs:

- "name": Name
- "value": TLV Blob

Key	Description	Possible Values
name	AID name.	String value. Application name, 10-32 ASCII hex characters representing 5-16 bytes. Example: "a0000000031010".
value	TLV Blob.	String Value. Application data in TLV format.

7.4. Beeper Settings

The fields below configure **Beeper** settings.

Key	Description	Possible Values
enabled	Enable or disables the Beeper.	String value: "true" or "false".
volume_pct	Volume determines the loudness of the beeper in a range from zero to 100%.	Decimal integer value from zero to 100 inclusive.
duration_ms	The length of time that the beep is turned on, in milliseconds.	Decimal integer value from zero to 65535 inclusive.
delay_ms	The delay before the beep starts, in milliseconds.	Decimal integer value from zero to 20000 inclusive.
frequency_hz	The audio pitch of the beep, in Hertz.	Decimal integer value from zero to 32768 inclusive; recommended value = 3000.

7.5. CAPK Settings

The fields below configure **CAPK** settings.

Key	Description	Possible Values
name	CAPK name.	5 bytes RID + 1 byte index. Example "a000000003ff"
hash_algorithm	Hash algorithm for the specified CAPK.	The only algorithm supported is SHA-1.The value is set to "01"
encryption_algorithm	Encryption algorithm for the specified CAPK.	The encryption algorithm in which this key is used. Currently support only one type: RSA. The value is set to "01"

hash_value	Hash value for the specified CAPK.	Calculated using SHA-1 over the following fields: RID & Index & Modulus & Exponen
exponent	Public Key Exponent	It can have two values: 3 (Format is "00000003"), or 65537 (Format is "00010001")
modulus_length	Modulus length.	LenL LenH Indicated the length of the next field. Example "5b01"
modulus	CAPK modulus for the specified CAPK.	This is the modulus field of the public key. Its length is specified in the field above.

7.6. ConfigGroup Settings

The fields below configure **ConfigGroup** settings.

Key	Description	Possible Values
group	Config group.	1 byte group number. Example "80".
data	Config group data.	Group data in TLV format.

7.7. ConfigMeta Settings

ConfigMeta are settings that are read from the device, or values calculated by the application, with the exception "memo", which can be user specified

Key	Description	Possible Values
type	Config file type.	"device"
production	If true, this is a production device.	Boolean value: "true" or "false".
customer	Customer info.	
id	Identifier; RFU.	0
notes	Notes for the config.	Application-generated notes.
version	Config file version.	"2.0.0"
terminal_type	Device for this config.	Device friendly name.
hash	Hash Value used to calculate integrity of the config file.	32-byte hash value.
memo	User-defined info.	Any string value user decides to include in this configuration file.
FirmwareVersion	Device firmware.	Firmware string.

7.8. CRL Settings

The fields below configure **CRL** settings.

Key	Description	Possible Values
version	Version number.	
number_of_records	Number of CRL records.	
size_of_records	Size of CRL records.	
crl_data	CRL Data.	[CRL] is 9 bytes: [5 bytes RID][1-byte CAPK Index][3 bytes serial number]

7.9. Customer Settings

The fields below configure **Customer** settings.

Key	Description	Possible Values
Company	Customer company.	User-entered string value.
Contact	Customer contact.	User-entered string value.
Id	Customer ID.	User-entered string value.

7.10. device_commands Settings

The fields below configure **device_commands** settings.

Key	Description	Possible Values
name	Command name.	Friendly name for reference. Example “Set Apple VAS”
command	Command data.	Example: “C766F53687089339205 53B7B9FFB16AEED9C77D5BFD9662 AF149A6B9F965B73F0CCA”
protocol	Command protocol.	• "IDG" • "NGA" • "ITP" • "RAW"
force_protocol_type	If true, ViVOconfig will attempt to write the command to the device even if the device uses a different protocol by default. If false or missing, ViVOconfig only writes commands to the device that match the default protocol.	Boolean value: “true” or “false”.

Key	Description	Possible Values
requires_KB	If true, the device only accepts the command while in KB mode. If false, the device accepts the command in HID mode.	Boolean value: “true” or “false”.
verify_only	If true, the command is a “get” command that only verifies a value from the device. If false, the command is a write command when writing the config to the device.	Boolean value: “true” or “false”.
verify_response	If the command is “verify_only”: “true”, this field contains the expected response for which to verify.	Boolean value: “true” or “false”.

7.11. Hardware Settings

The fields below configure settings.

Key	Description	Possible Values
product_type	Device type.	String value. Device dependent.
processor_type	Device processor type.	String value. Device dependent.
hardware_ver	Device hardware version.	String value. Device dependent.
serial_num	Device serial number.	String value. Device dependent.
firmware_ver	Device firmware version.	String value. Device dependent.
sams_cnt	Device SAM count.	String value. Device dependent.
mcu_uid		String value. Device dependent.
contactless_available	If true, device can use contactless.	Boolean value: “true” or “false”.

7.12. install_rules Settings

The fields below configure **install_rules** settings.

Key	Description	Possible Values
finalize_in_kb_mode	If true, ViVOconfig puts the device into in KB mode at the end of installation.	Boolean value: “true” or “false”.
keep_all_Contact_AID	If true, ViVOconfig keeps existing Contact AIDs instead of erasing them when writing new ones.	Boolean value: “true” or “false”.

keep_all_Contact_CAPK	If true, ViVOconfig keeps existing Contact CAPKs instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_Contact_CRL	If true, ViVOconfig keeps existing Contact CRLs instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_Contactless_AID	If true, ViVOconfig keeps existing Contactless AIDs instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_Contactless_CAPK	If true, ViVOconfig keeps existing Contactless CAPKs instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_Configuration_Groups	If true, ViVOconfig keeps existing Configuration Groups instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_terminal_data	If true, ViVOconfig keeps existing terminal settings instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
keep_all_images	If true, ViVOconfig keeps existing images instead of erasing them when writing new ones.	Boolean value: “true” or “false”.
bypass_Contactless_reset	If true, ViVOconfig does not execute a CTLS reset 04-09 command at the start of writing a new config.	Boolean value: “true” or “false”.
Set_DFED22	If true, ViVOconfig customizes tag DFED22 in terminal settings to be <Serial_Number><Kernel Version>.	Boolean value: “true” or “false”.
Set_9F1E	If true, ViVOconfig customizes tag 9F1E in terminal settings to be <Serial_Number>.	Boolean value: “true” or “false”.
Set_Self_Check_Time	If value set to “HHMM”, this sets the device self-check time to that hour/minute value.	“1300”
Set_Date_Time	If true, ViVOconfig attempts to set the device date_time to match the system clock.	Boolean value: “true” or “false”.
Load_Images	If this setting contains a directory name, ViVOconfig attempts to load any images it finds there. That directory must be located at the same level as the configuration file.	“images”

7.13. Msr_Setting Settings

The fields below configure **Msr_Setting** settings.

Key	Description	Possible Values
function_id	MSR function.	1 byte as string. Example "2e".
name	Friendly name.	Example: "Polling Mode".
value	MSR function setting.	1 or more bytes as string. Example: "0d".

7.14. Encryption Settings

The fields below configure **Encryption** settings.

Key	Description	Possible Values
data_encryption_type	Key format.	• "TDES" • "AES" • "NONE"
data-encryption_variant	Key type for ICC DUKPT.	• "Data" • "PIN" • "NONE"
msr_encryption_enables	MSR encryption is enabled.	Boolean value: "true" or "false".
TransArmor TID	TransArmor ID.	ID in string format.

7.15. SmartCard_Setting Settings

The fields below configure **SmartCard_Setting** settings.

Key	Description	Possible Values
function_id	MSR function.	1 byte as string. Example "2e".
name	Friendly name.	Example: "Polling Mode".
value	MSR function setting.	1 or more bytes as string. Example: "0d".

7.16. Terminal Settings

The fields below configure **Terminal** settings.

Key	Description	Possible Values
checksum	EMV kernel checksum.	String hash value.
configuration	EMV kernel configuration number.	String value. Example "2".
data	EMV kernel terminal data.	String value. EMV tags in TLV format.

7.17. TerminalInfo Settings

The fields below configure **TerminalInfo** settings.

Key	Description	Possible Values
module_ver_info	Device module version information.	String value. Device dependent.
firmware_ver	Device firmware version.	String value. Device dependent.
contact_emv_kernel_ver	Contact EMV kernel version.	String value. Device dependent.
contact_emv_kernel_checksum	Contact EMV kernel checksum.	String value. Device dependent.
contact_emv_kernel_configuration_checksum	Contact EMV kernel configuration checksum.	String value. Device dependent.
usb_bootloader_version	USB bootloader version.	String value. Device dependent.
quickchip_mode	If true, quickchip mode is enabled for the device.	String value: "true" or "false".
mode	HID or KB.	• "HID" • "KB"
pmc_status source_01-05	Value to set for command 01-05 Get PMC Status.	String value representing the command. Example "0905".
fileDirectory	Directory Listing. Lists all the images on the device, if any.	String value representing the file directory information.