Programming Job Manager User Guide

Libero SoC v11.8 SP3

Microsemi
Power Matters.™

Table of Contents

Introduction

The Job Manager tool is a part of the SPPS (Secured Production Programming Solution) ecosystem that consists of the Libero, Job Manager, and FlashPro Express applications as well as User and Manufacturer HSM servers (U-HSM and M-HSM, respectively).

This user guide describes how to use the Job Manager to organize manufacturing flow as described in the SPPS User Guide.

The Job Manager is primarily intended for use by the OE (Operation Engineer) who is responsible for the manufacturing process organization and security.

The current version of the Job Manager is based on the Tcl interface running in command line mode.

The Job Manager supports the creation of:

- Programming Jobs for the regular production programming flow (non-HSM manufacturing flow in this document)
- Programming Jobs for the secured production programming flow (HSM-based manufacturing flow in this document)
- Bitstream files in various formats (STAPL, DAT, SPI, and so on) for use by third party programming tools (non-HSM flow)

Programming Jobs created by the Job Manager can be programmed into the device using FlashPro Express (see the FlashPro Express User's Guide) or IHP (In House Programming).

Programming bitstream generation supported by the Job Manager is based on design information imported from Libero. eNVM data and security settings can be modified in the Job Manager project, which allows update of areas such as M3 firmware image and enforcement of security policies.

The Job Manager allows generation of the programming bitstreams outside the Libero flow, eliminating the need for the OE to handle the design using Libero, obtain design level Libero licenses, and address design migration to newer versions of Libero.

HSM-based Manufacturing Flow

The HSM-based manufacturing process uses device-supported security protocols. For more information, refer to the SPPS User Guide.

The U-HSM allows the user to generate and use various encryption and pass keys. Cryptographical operations requiring those keys are executed inside the security boundaries of the HSM Module:

Operation support for the HSM-based Manufacturing Flow is as follows:

1. Key import, generation, and use under protection of the U-HSM and the M-HSM
2. Initial secure key injection into the device using the Authorization Code protocol
3. Secure data transmission between the U-HSM and the M-HSM
4. Overbuild protection
5. Device authentication
6. U-HSM verification of cryptographically sealed certificate of conformance (CoC) from the design programmed into the device
7. U-HSM verification programming job end certificates
8. Master and Upgrade programming job types
9. eNVM Update
10. eNVM client selection
11. Security overwrite

Non-HSM Manufacturing Flow

Non-HSM Manufacturing Flow support by the Job Manager allows bitstream file and programming job generation outside of the Libero tool. The Job Manager supports the following operations for non-HSM Manufacturing Flow:

1. Initial key loading via KLK-protected bitstreams
2. Generate UEK1/UEK2/UEK3 encrypted update bitstreams
3. eNVM Update
4. eNVM Client selection
5. Key value overwrite
6. Security overwrite

HSM Server Requirements

In the HSM-based Manufacturing Flow, the Job Manager is configured to work with the U-HSM to generate the HSM Programming Job, and HSM Programming Jobs require the M-HSM for job execution. For more information about HSM servers, refer to the Secure Production Programming Solution (SPPS) User Guide. For information about installation, refer to the User HSM Installation and Setup Guide for the U-HSM and the Manufacturing HSM Installation and Setup Guide for the M-HSM.

Terms and Definitions

This user guide uses the following terms:

Term	Definition
U-HSM	User HSM
M-HSM	Manufacturing HSM
OE	Operation Engineer
DSN	Device Serial Number
IHP	In House Programming
JDC	Job Data Container file
SPM	Security Policy Manager

Referenced Documents

This user guide references the following documents:

- *Secure Production Programming Solution (SPPS) User Guide* (Microsemi SOC)
- *Libero User's Guide* (Microsemi SOC)
- *FlashPro Express User's Guide* (Microsemi SOC)
- *User HSM Installation and Setup Guide* (Microsemi SOC)
- *Manufacturer HSM Installation and Setup Guide* (Microsemi SOC)

1 – HSM Parameter Configuration

To use Job Manager in the HSM flow, U-HSM parameters must be set. These parameters are stored in the user level DEF file and automatically loaded for any new or existing Job Manager project.

U-HSM configuration data specifies:

- IP address of the U-HSM server
- U-HSM UUID assigned by Microsemi
- U-HSM Master Key UUID
- Default location of the keyset repository (see the Libero User's Guide)
- M-HSM UUID assigned by Microsemi

The set_hsm_params Tcl command is used to configure HSM. See "Tcl Interface" on page 15 for details.

2 – Keyset File

The keyset file is used in the HSM flow only. It contains U-HSM generated keys encrypted with the Master Key of the U-HSM. New keyset files can be generated randomly by the HSM, derived from the existing keyset files, or created from the imported plain text values.

Keyset files are stored in the keyset repository, and can be shared among different Job Manager projects. The repository is configured via HSM parameters (refer to the FlashPro Express User's Guide).

Keyset files contain the following keys:

- Ticket Key that encrypts all keys in the keyset file.
  - Ticket Key is protected by the HSM Master Key.
  - Keyset file contains HSM Master Key UUID to help identify the origin of the file.
- Encryption keys: UEK1, UEK2, UEK3
- Pass keys: UPK1, UPK2, DPK
- Base keys for deriving per-device key values for UEK1/UEK2/UEK3/UPK1/UPK2/DPK

For more information about SPPS key management, refer to the Secure Production Programming Solution (SPPS) User Guide.

A keyset file is generated outside the Job Manager project and is associated upon creation of a new Programming Data entry (refer to the User HSM Installation and Setup Guide).

The create_keyset Tcl command is used to manage keyset files. See "Tcl Interface" on page 15 for more information and keyset generation scenarios.

3 – Programming Data

Programming Data entry is created from the design information imported from Libero in a JDC (Job Data Container) file (see the Libero User's Guide for details about JDC Export for a Libero project). Programming Data contains all information required for bitstream generation in the HSM flow and non-HSM flow. A bitstream that is initialized for non-HSM flow can be exported directly from Programming Data into a bitstream file in a format selected by the user.

The following design data modifications can be done by the user within Programming Data:

- One or more eNVM clients can be updated with an image loaded from external data files.
- Design security can be overwritten from the external SPM (Security Policy Manager) file from Libero.
- In non-HSM flow, plain text values of the encryption and pass keys can be changed. In the HSM flow, all key values are used from the HSM-protected keyset file.

A Job Manager project can have one or many Programming Data entries to support programming of multiple Microsemi devices on the same board.

The following sections provide information about the creation and modification of data in Programming Data entry.

Create Programming Data from JDC File

New Programming Data is created using the new_prog_data Tcl command. It must be created within an existing or new Job Manager project.

When creating a new Programming Data entry, design data is copied from the external JDC file within the current project. After this step, the external JDC file is no longer used.

In the HSM flow, all keys are generated and controlled by the U-HSM. Therefore, in this flow new Programming Data must be associated with a keyset file created as shown in "Keyset File" on page 6. All cryptographic operations involving protected keys are executed inside the HSM module. For example, a Master bitstream is generated using the U-HSM and programmed via the Authorization Code protocol using the M-HSM.

eNVM Update

This is an optional step that allows the user to modify one or more eNVM clients found in the design loaded into the Programing Data. Refer to the Secure Production Programming Solution (SPPS) User Guide for the use model definition.

To add a client update, the design in the Programming Data entry must have an eNVM component that already contains a target eNVM client. The size of the update data must be equal to or smaller than the client size. See the set_envm_update Tcl command in "Tcl Interface" on page 15 for details.

An existing eNVM client update can be removed using the remove_envm_update Tcl command.

Key Overwrite (Non-HSM Flow)

This feature applies to the non-HSM flow and allows the user to modify plain text key values for the keys coming from Libero as a part of the design security settings. New key values can be set for UEK1/UEK2/UEK3/UPK1/UPK2/DPK using the set_key Tcl command. Key overwrites can be reverted using the remove_key Tcl command.

Security Overwrite

Security overwrite ignores security settings imported into the Programming Data entry from Libero (via the JDC file) and uses settings imported from the SPM file on disk. After import, the external SPM file is no longer used by the Job Manager.

Security overwrite is available in HSM flow and non-HSM flow. In both flows, security policy is used from the overwrite, while key values follow these rules:

HSM Flow

- Key values are always used from the keyset file.

Non-HSM Flow:

- Key values are used from the security settings of the security overwrite.
- For keys that were overwritten using the set_key command (see "Key Overwrite" on page 14), the key overwrite value is used.

For details, see the set_security_overwrite and remove_security_overwrite Tcl commands in "Tcl Interface" on page 15.

Bitstream Initialization

Bitstream initialization allows the user to set up bitstream generation parameters for use during:

- HSM job export
- Non-HSM job export
- Export of a programming bitstream in non-HSM flow

The bitstream can be initialized for use in the HSM programming flow and non-HSM programming flow. For use model details about HSM and non-HSM bitstreams, refer to the Secure Production Programming Solution (SPPS) User Guide.

Programming Data can have one or more bitstream entries that can be used by the same or different programing jobs.

Bitstream initialization is done with the init_bitstream Tcl command.

HSM Flow

The following sections describe how to initialize and use programming bitstreams for various situations in the HSM flow. Because all keys in the HSM flow are protected by the HSM, the U-HSM must generate bitstreams for all cases described below.

Master Bitstream

The Master bitstream in HSM is designed to program initial security and all other user-selected device features in an untrusted environment. Secure key loading is achieved using the device-supported Authorization Code protocol. Refer to the Secure Production Programming Solution (SPPS) User Guide for more information.

The Master bitstream can be generated to program project or per-device UEK1/UPK1/UEK2/UPK2/UEK3/DPK values.

Project keys are inserted into the bitstream from the keyset file upon bitstream generation.

Per-device keys are generated and infused into the programming bitstream by the M-HSM during device programming. Per-device key value is derived from respective base keys in the keyset file and device DSN (serial number). Per-device protocol and key types are specified with the init_bitstream Tcl command parameters.

The Master bitstream programs security settings imported into Programming Data from Libero or according to SPM overwrite, if any.

Important! Due to security policy, after the initial key loading, programming actions such as ERASE and VERIFY require HSM support to unlock device security.

Warning! Security settings programmed into the device can only be changed with the ERASE action. The ERASE action does not erase content of the eNVM. eNVM memory is fully accessible after security settings have been erased.

UEK1/UEK2/UEK3 Update Bitstream

This type of bitstream is used for reprogramming Fabric and/or eNVM. The security component cannot be reprogrammed with this file type. This bitstream can be used if the device already has security programmed.

UEK1/UEK2/UEK3 project keys, no security lock

In this case, all devices in the project have the same UEK1, UEK2, or UEK3 values, and target device feature programming is allowed without FlashLock/UPK1 match. The Job Manager can generate a standalone programming file or a non-HSM programming job that does not require the M-HSM during programming.

UEK1/UEK2/UEK3 project keys, security locked

If the target device feature programming is locked, the M-HSM must perform a secured unlock of the device, because the plain text value of the lock key cannot be used in an untrusted environment. This type of bitstream can be used in an HSM programming job. UPK1 unlock is performed securely via the OTPK protocol. For more information, refer to the Secure Production Programming Solution (SPPS) User Guide.

UEK1/UEK2/UEK3 per-device keys, no security locks, DSN is known

If UEK1, UEK2, or UEK3 are per-device keys, target features are not locked, and DSN for the target device(s) is known, the user has an option to generate a device-specific programming bitstream that does not require the M-HSM during production. The bitstream can be exported as a standalone bitstream file or a non-HSM programming job. In either case, DSN must be provided in the export_bitstream Tcl command during bitstream file generation or in the add_microsemi_device Tcl command that adds target devices to the chain inside the programming job.

UEK1/UEK2/UEK3 per-device, all other cases

For all other cases related to per-device UEK1/UEK2/UEK3, the M-HSM and HSM programming job must be used. If target device features are locked by per-device UPK1, UPK1 unlock is performed securely via the OTPK protocol. For more information, refer to the Secure Production Programming Solution (SPPS) User Guide.

Non-HSM Flow

In non-HSM flow, the keyset file is not used. All key values are used from the Libero design security setting. There are two mechanisms to overwrite the user-defined design security keys (UPK1, UPK2, UEK1, UEK2, UEK3, and DPK):

1. Security Overwrite—The security setting and key values set supersede the original Libero settings.
2. Key Overwrite—The key values set supersede both the Libero and Security Overwrite setting.

The following sections describe non-HSM bitstream types that can be generated for production programming in a trusted environment.

Trusted Facility Bitstream

This bitstream type can program Fabric and/or eNVM. The entire bitstream is encrypted with the KLK encryption key.

Master Bitstream

Similar to Trusted Facility, but also programs security. After custom security is programmed, all Microsemi factory default key modes, including KLK, DFK, KFP, and KFPE key modes, become disabled.

Note: Per security policy, programming of UEK1 or UEK2 will program the UPK1 or UPK2 passkeys, respectively, and lock the security segment. As a result, the ERASE and VERIFY actions in generated bitstream files or programming jobs will contain plain text UPK1/UPK2 values. This is required to unlock security segments for the programming actions when using the non-HSM flow (the HSM-based flow uses encrypted one-time passcodes).

UEK1/UEK2/UEK3 Update Bitstream

This bitstream type can reprogram Fabric and/or eNVM device features. If the target device programming is protected by FlashLock/UPK1, plain text values of UPK1 are included in the exported bitstream file/ programming job. when using the non-HSM flow.

Export Programming Bitstream File

Export of the programming bitstream file is available in non-HSM flow only. Export is handled by the export_bitstream_file Tcl command and can be performed in all supported programming file types.

Specifying the optional DSN parameter is applicable only for situations explained in "Non-HSM Flow".

The exported bitstream file is created in a user-specified location.

4 – Programming Job

A programming Job is a set of data used by programming systems for device programming in HSM and non-HSM flows.

The current SPPS ecosystem supports FlashPro Express and IHP job types.

FlashPro Express can program HSM and non-HSM jobs (refer to the FlashPro Express User's Guide), and IHP supports the HSM job type only.

A programming job contains the following data:

- Job type (FlashPro Express or IHP)
- Job origin
- Bitstream(s) for various programming actions (PROGRAM, ERASE, and VERIFY)
- Hardware setup information – for FlashPro Express job type
  - Type of hardware interface (JTAG in this version of the Job Manager)
  - Configuration
- Job device(s) – includes basic device information
- Data for HSM flow only
  - M-HSM UUID
  - U-HSM UUID
  - Encrypted Job Tickets authorizing programming actions and overbuild protection under control of the HSM
  - Encrypted keys and security protocol data required by HSM protocols

HSM Programming Job

A Programming Job can be set up for execution in an untrusted environment. In this case, the M-HSM protects cryptographic key material and other sensitive data required by device security protocols during the manufacturing process.

HSM data is added to the Programming Job with HSM Task(s). See "HSM Tasks" on page 12 for details.

A Programming Data entry is created in the Job Manager project using the new_prog_job Tcl command.

Creating a FlashPro Express Programming Job

After creating a FlashPro Express Programming Job, the user specifies type of the hardware setup:

- JTAG Chain – supported in the current release of the Job Manager

Configuring a JTAG Chain

The following device types can be added to a JTAG chain:

- Microsemi device targeted for programming
- Microsemi bypass device not targeted for programming
- Non-Microsemi bypass device

Each device has a user-defined name that is unique within the JTAG chain.

Microsemi devices can be programmed using a bitstream generated by a Programming Data entry or by using existing programming bitstream files (STAPL) loaded from disk.

Adding a Microsemi Device for Programming by Generated Bitstream

If a Microsemi device is programmed by a bitstream generated from Programming Data, the bitstream must be specified in the Programming Data and bitstream name parameters in the add_microsemi_prog_device Tcl command (see "Tcl Interface" on page 15 for details).

The "DSN" parameter is used in the HSM flow to create a device-specific update programming file (see "UEK1/UEK2/UEK3 Update Bitstream" on page 10 for details). Actual bitstream generation occurs during export of the programming job (see "HSM Task Export" on page 13).

Adding a Microsemi Device for Programming by Existing Bitstream File

A Microsemi device can be programmed using the existing bitstream file generated outside the Job Manager.

Note: If an external bitstream file has been loaded in the device, it cannot be programmed using HSM, and tickets cannot be created.

Use the add_microsemi_prog_device Tcl command pointing to the target bitstream file on disk specifying the path to the file with the "bitstream_file" parameter.

Adding a Microsemi Bypass Device

A Microsemi bypass device can be added by specifying the device name or pointing to the device programming file.

Refer to the add_microsemi_bypass_device Tcl command for more information.

Adding a Non-Microsemi Bypass Device

A non-Microsemi device can be added to the JTAG chain with the add_non_microsemi_bypass_device Tcl command.

JTAG bypass parameters can be specified either by pointing to the BSDL file accepted by the command or by explicit parameter specification. Refer to the add_non_microsemi_bypass_device Tcl command for more information.

Export of Non-HSM Programming Job

A non-HSM Programming Job is exported from the Job Manager with the export_prog_job Tcl command. All bitstreams generated from Programming Data entries are created during command execution.

Note: A Programming Job that has one or more HSM Tasks is considered to be HSM type and cannot be exported using the export_prog_job command. See "HSM Task Export" on page 13 for more information.

HSM Tasks

The HSM task in the HSM flow allows flexibility in organizing the manufacturing process. It is possible to utilize multiple Contract Manufacturers simultaneously, or the entire manufacturing volume can be split onto smaller chunks for overbuild protection. For example, after creating a Programming Job, the OE can create and export an HSM Task for each manufacturer in production.

HSM Tasks add HSM data to the Programming Job. For each HSM Task, the user creates Job Tickets and specifies programming actions for each ticket. Overbuild protection and other protocol-specific information is specified during ticket creation.

For more information about the HSM use model and flow description, refer to the Secure Production Programming Solution (SPPS) User Guide.

Job Tickets

The HSM Task Ticket (Job Ticket in this document) is used to enforce security policies on the manufacturing side and encrypt sensitive information used by device security protocols.

A Job Ticket is created per device in the Programming Job. Each device can have one or more tickets. The Job Ticket is created per the user-selected programming action.

A new Job Ticket is created with the new_hsmtask_ticket Tcl command. The "max_device" parameter is used to limit the number of devices a programming action can be executed on.

Job Request

A Job Request is exported from the Job Manager Project after creation of all tickets within the HSM Task. The Job Request is then sent to and processed by FlashPro Express or IHP using its M-HSM.

A Job Request is created with the hsmtask_m_request Tcl command.

Job Reply

A Job Reply returns ticket generation information created by the FlashPro Express/IHP. This information is cryptographically bound to the physical M-HSM/U-HSM module that processed the Job Request. After performing this handshake protocol, the HSM Job exported from this HSM Task can only be used with that particular module. This prevents HSM Task replication on the manufacturing side.

A Job Reply is generated by FlashPro Express or IHP and can be imported into the requesting U-HSM Task with the hsmtask_m_reply Tcl command.

HSM Task Export

An HSM Task (HSM Job in this document) can be exported with the export_hsmtask Tcl command. This command executes the part of export done during the non-HSM job export and adds HSM-specific information to the job export container. This data includes job tickets, encryption keys, protocol data, and other HSM-specific information. The HSM job can only be exported after importing the Job Reply.

Job Status

Job Status can be generated by FlashPro Express or IHP during job execution or after ending the job. A Job Status file is generated and sent to the customer.

An HSM Programing Job being executed on the manufacturing side can be ended when all target devices are programmed, or the job can be terminated at any time.

The Job Manager uses Job Status to:

- Validate job end status, which is cryptographically protected proof that the job has ended and can no longer execute programming actions controlled by its tickets
- Display the number of devices that can be handled by each ticket
- Ensure that the correct bitstream is programmed into each device by validating the CoCs.

5 – Export SPI Directory

The Job Manager can be used to create the SPI directory used by SmartFusion2/IGLOO2 devices during auto-update and programming recovery. The SPI directory contains information about golden and update programming bitstreams placed by the user into SPI flash.

The SPI directory is created with the export_spi_directory Tcl command. It does not require an existing Job Manager project.

The version number for the golden and update bitstreams can be entered manually or by providing previously generated SPI files. The user also needs to provide addresses for both images in the final flash memory.

For more information about using the SPI flash directory, refer to the Libero User's Guide.

6 – Tcl Interface

Application

```
set_hsm_params -hsm_server_name <hsm_server> -u_hsm_uuid <u_uuid> -
u_master_hsm_uuid <u_master_uuid > -hsm_key_set_dir <keyset_dir> -
m_hsm_uuid <m_uuid>
```

hsm_server

Name or IP address of HSM server machine.

u_uuid

User HSM UUID.

u_master_uuid

User HSM Master UUID.

keyset_dir

Keyset repository location: a directory in which the keyset files will be created or used.

m_uuid

Manufacturer HSM UUID.

This command saves the HSM parameters for the Job Manager application. This remains in effect until its overridden using this same command.

```
get_software_info [-version]
```

version

Get the software version info.

This command prints the Job Manager software information.

Keyset Management

```
create_keyset -file <output_file_name> [-source_file
<source_file_name>] [-kip <token_key>]
[-upk1 <token_key>] [-upk1_base <base_key>]
[-uek1 <token_key>] [-uek1_base <base_key>]
[-upk2 <token_key>] [-upk2_base <base_key>]
[-uek2 <token_key>]  [-uek2_base <base_key>] [-dpk <token_key>]
[-dpk_base <base_key>]
[-uek3 <token_key>] [-uek3_base <base_key>]
```

output_file_name

Name of the new keyset file. Name only, no path. The file is created in the keyset repository defined by the Job Manager application settings.

source_file_name

Optional source keyset file name for key import. The ticket key in this file must be encrypted with the same U-HSM Master Key.

token_key

Optional value of the token keys that will be imported into the new keyset. This parameter takes precedence over the keys in the source file, if specified. Token key values can only be specified in plain text format.

base_key

Optional value of the base keys (i.e., keys used to derive device-specific token keys) that will be imported into the new keyset. This parameter takes precedence over the keys in the source file, if specified. Base key values can only be specified in plain text

This command creates a new keyset file for the HSM flow tasks. It can create keys for the following scenarios:

- Create a new keyset file
  - All keys are randomly generated by the U-HSM
- Create a new keyset file with key import
  - Ticket key is randomly generated
  - User-specified keys are imported and protected by the ticket key
  - The rest the keys are randomly generated and protected by the ticket key
- Create a modified copy of the existing keyset file
  - New keyset file receives copies of the keys from the source file including ticket key
- Ticket is protected by the same U-HSM Master Key
  - User-specified keys will be imported into the new keyset file in place of the keys in the source keyset file. All imported keys are protected by the ticket key from the source file

Notes:

1. Because the same keyset file can be shared between different Job Manager projects, create_keyset always creates a new file. It cannot delete, rename, or overwrite existing keyset files. All such operations should be handled by the user manually.
2. Keys specified in the keyset file always take effect regardless of whether 'set_security_overwrite' command is run.
3. UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

Project Management

new_project -location <path> -name <file_name>

path

Top level project directory.

file_name

Project file name.

This command creates a new Job Manager project. The project directory can be moved to any other location on disk as project internally uses only relative paths. If the project already exists, command will exit with error.

open_project -project <path>

path

Path to the Job Manager project file (.jprj).

This command opens an existing Job Manager project.

close_project [-save <TRUE |FALSE>]

This command closes the Job Manager project with or without saving it. If project was modified, then -save option must be specified.

Programming Data

Design Import

```
new_prog_data -data_name <name>
              -import_file <path>
             [-hsm_key_set < keyset_name>]
```

name

Programming data entry name.

path

Libero design data file (JDC).

keyset_name

Name of the keyset file. File name only. The file will be in the keyset directory specified by the Application settings.

This command creates a new Programming Data entry from JDC (design data exported from Libero project). If keyset parameter is specified, all the key management is handled by the HSM.

Security Modifications

```
set_key -data_name <name>
[-upk1 <upk1_value>] [-uek1 <uek1_value>] [-upk2 <upk2_value>] [-uek2
<uek2_value>] [-dpk <dpk_value>] [-uek3 <uek3_value>]
```

name

Name of the Programming Data.

*uek*_value*

New value for the selected encryption key.

*upk*_value*

New value for the selected pass key.

dpk_value

New value for the DPK.

This command overwrites key values imported from JDC with the ones specified in the command arguments.

Note:

1. This command is applicable to non-HSM flow only.
2. Keys specified by this command always take affect irrespective of whether 'set_security_overwrite' command is run.
3. UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

remove_key -data_name <name> -key_names <ALL UPK1 UEK1 UPK2 UEK2 DPK> UEK3

name

Name of the Programming Data.

key_names

Names of the keys whose value is to be reverted (can specify multiple values).

This command reverts the value of the specified encryption and/or pass key to the value imported from JDC. This command supports action opposite to set_key.

Note: This command is applicable to non-HSM flow only.

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

set_security_overwrite -data_name <name> -file <spm_file_path>

name

Name of the Programming Data entry.

spm_file_path

File path of new SPM file.

This command overwrites security settings in a design imported from Libero with the new SPM file. This command is applicable to HSM and non-HSM flows. The way security policies and keys are overridden are explained below:

- Security policies: Always overwritten from the new SPM file (HSM and non-HSM flows)
- Security Keys (UPK1, UEK1, UPK2, UEK2, UEK3, DPK)
  - Non-HSM flow:
    - If set_key has already been run before or is run after this command, then all keys specified in set_key are used. If any keys are not specified in set_key, they are taken from the new SPM file specified as argument.
    - If set_key has not been run, then all security keys are taken from new SPM file.
  - HSM flow:
    - Security keys are always taken from the keyset file.

Note: UEK3 is only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

remove_security_overwrite -data_name <name>

name

Name of the Programming Data entry.

This command removes existing security overwrite settings. This command supports action opposite to set_security_overwrite. Note that this command will fail if set_security_overwrite has not been run previously.

eNVM Client Modifications

```
set_envm_update -data_name <data>
                -client_name <client>
                -file < file_path>
                [-overwrite <YES | NO>]
```

data

Programming data entry name.

client

Name of the target eNVM client.

file_path

File path to the client update data file.

This command creates a new client update entry in the specified Programming Data. If an update already exists for that client, the command will fail unless the optional -overwrite parameter is set to "yes". Data file size must not exceed the client size defined by the Libero project and imported via the JDC file. This command allows the use of all eNVM data file formats supported by Libero (refer to the Libero User's Guide for details).

```
remove_envm_update -data_name< data > -client_name < client > -all <YES
| NO>
```

data

Programming data entry name.

client

Name of the target eNVM client.

all

Removes all clients. Default is "YES"

This command removes an entry that was created by set_envm_update. All client updates can be removed using "all" key set to "YES". A removed client update results in the client being reverted to the original eNVM data received in the JDC.

Bitstream Management

```
init_bitstream -data_name <data_name>
                          -bitstream_name <file_name>
                          -bitstream_type <bitstream_type>
                           [-features <feature_selection>
                           [-use_protocol <hsm_protocol_selection>
                               <hsm_protocol_parameters>]
                           [-envm_clients <client_name>]
                           [-generate_cofc]
                           [-auth_keymode <keymode>]
```

data_name

Name of the Programming Data entry.

bitstream_name

Name of the bitstream being added (without path or extension).

bitstream_type

Specifies bitstream type. Resulting bitstream will be generated accordingly to the security policy specified in Programming Data security settings. Only one bitstream type can be selected at a time:

TRUSTED_FACILITY

Non-HSM flow only: programs selected bitstream components (fabric/eNVM) using KLK key mode.

MASTER

Programs security and any other selected components.

Non-HSM flow: Uses KLK keymode to program security and other components. If security is programmed with UEK1/UEK2, then bitstream will also include plaintext UPK1/UPK2 respectively in order to enable erase and verify actions.

HSM flow: Uses DFK, KFP, or KFPE keymodes to program security and other components. In this case all keys are encrypted by HSM.

UEK1

This command programs (updates) selected bitstream component(s) (fabric/eNVM) using UEK1 key mode. If the target component is update-protected, UPK1 will be used to unlock target component. In this case, for the non-HSM flow generated bitstream uses plain text UPK1 value unlocking security. In the HSM flow (i.e., if Programming Data uses the keyset file to protect keys), the M-HSM performs security unlock using the encrypted value of UPK1.

UEK2

Similar to UEK1, but uses UEK2 and UPK2 respectively.

UEK3

Similar to UEK1, but uses UEK3.

feature_selection

Any combination of the features or ALL (default option)

SECURITY

Security is selected for programming.

Note: Security is always programmed in the MASTER bitstream.

FABRIC

FPGA fabric is selected for programming.

ENVM

eNVM is selected for programming.

ALL

All features available in Design Data to be selected (default).

hsm_protocol_selection

Allows selecting one of the following HSM security protocols:

AUTH_CODE

Securely sends KIP to the device using Authorization code. Use this option when all the user keys are project keys. This protocol supports MASTER bitstream flow and UEK1/UEK2/UEK3 update bitstreams flow.

UNIQUE_KEY

Allows programming of unique per-device keys in the Master flow. For the UEK1/UEK2/UEK3 update flow this option can be used to generate per-device bitstream files, if DSN is known at the time of job or bitstream file generation or to use per-device keys during programming jobs using the M-HSM.

protocol_parameters

-unique_key_types <UEK1 UEK2 UPK1 UPK2 DPK UEK3>

Parameter required for UNIQUE_KEY protocol. It specifies which of the user keys are per-device keys. Each per-device key is derived from a base key and DSN during programming, which makes it device-specific.

generate_cofc

Parameter to setup the generation and export of Certificate of Conformance.

This command allows the user to set up parameters for bitstream generation. Actual generation occurs upon job or bitstream file export. Bitstream generation is based on design information in the Programming Data entry, including all modifications such as eNVM update, security overwrites, etc.

auth_keymode

Optional parameter that specifies what keymode to use for Authorization component in MASTER bitstream HSM flow. The table below shows the valid and default value for this parameter.

Device	Valid Values	Default Value
M2S060, M2GL060, M2S090, M2GL090, M2S150, M2GL150	DFK, KFP, KFPE	KPFE
Other SmartFusion2 and IGLOO2 devices	DFK	DFK

Note: UEK3, KFP and KFPE are only available for M2S060, M2GL060, M2S090, M2GL090, M2S150, and M2GL150 devices.

```
export_bitstream_file -data_name <name>
                      -bitstream_name <bitstream_name>
                      -formats <format_selection>
                      -export_path <path >
                      [-dsn <dsn_value>]
```

name

Name of the programming data.

bitstream_name

Name of the bitstream entry to be exported.

format_selection

Any combination of the format types. File will be created for each selected type:

STAPL STAPL file (non-HSM and HSM Update flow)

SPI SPI file (non-HSM and HSM Update flow)

DAT DAT file (non-HSM and HSM Update flow)

path

Full file path without any extension. Extensions are determined based on formats parameter above.

dsn

DSN of the device for which to generate per-device UEK1/2/3 bitstream files.

This command generates and exports bitstream file(s) of the type determined by the formats parameter. File generation is based on the information in the Programming Data entry.

Programming Job

```
new_prog_job -job_name <name> -job_type <job_type> [-setup <
hw_setup_type>]
```

name

Name of the programming job entry.

job_type

Type of Job. One of the following:

- IHP: Job for IHP flow.
- FPExpress: Job for FPExpress flow.

hw_setup_type

Type of the HW setup.

- JTAG_CHAIN: JTAG chain (default parameter)

This command creates a new Programming Job. The Programming Job can be for IHP flow or for FPExpress. The job is created for a specific hardware setup type. The default setup type is JTAG chain. Job name must be unique among other job names. The job can have one or more devices and optional HSM Tasks for the HSM flow.

```
add_microsemi_prog_device -job_name <job>
      -device_name <device>
      -device_hw_location <location>
      [-data_name <data>]
      [-bitstream_name <bitstream>]
      [-dsn <dsn_value>]
      [-bitstream_file <bitstream file>]
```

job

Name of the Programming Job to add device.

device

User name of the device. Must be unique within the Job.

location

Hardware location of the device in the setup: Chain index for JTAG chain.

data

Name of the Programming Data containing bitstream file.

bitstream

Bitstream name in the Programming Data.

dsn

Parameter to set the DSN of the device. This is used for generating per-device UEK1/2/3 bitstream files.

bitstream_file

Path of STAPL file to program this device.

This command manually adds a Microsemi device targeted for programming by a bitstream generated from the specific design (Programming Data) or STAPL file. Device name must be unique among other devices inside specified jobs.

Note: You can add any valid Microsemi device using a standalone STAPL file. However, only SmartFusion2 and IGLOO2 are supported when using the Programming Data option.

```
add_microsemi_bypass_device -job_name <job>
                            -device_name <device>
                            -device_hw_location <location>
                            [-device_type <die_name>]
                            [-bitstream_file <bitstream_file>]
```

job

Name of the Programming Job to add device.

device

User name of the device. Must be unique within the Job.

location

Hardware location of the device in the setup: Chain index for JTAG chain.

die_name

Name of Microsemi device (eg: M2S010, M2GL090TS, etc.,)

bitstream_file

STAPL file path for this device.

This command adds a Microsemi bypass device by either specifying the die name or by specifying a STAPL file.

```
add_non_microsemi_bypass_device -job_name <job>
                                -device_name <device>
                                -device_hw_location <location>
                                [-ir <IR_len>]
                                [- tck <tck>]
                                [-file <bsdl_file>]
```

job

Name of the Programming Job to add device.

device

User name of the device. Must be unique within the Job.

location

Hardware location of the device in the setup: Chain index for JTAG chain.

ir

IR length.

tck

Max TCK frequency (in MHz).

bsdl_file

BSDL file path for targeted non-Microsemi device.

This command adds a non-Microsemi bypass device by either specifying IR length and TCK frequency or by specifying BSDL file.

```
export_prog_job -job_name <job>
                -location <path>
                -name <file>
```

job

Name of the Programming Job being exported.

path

Path to the directory in which Job file will be exported.

name

File name of job file to be exported.

This command exports a non-HSM job for FlashPro Express or IHP.

Note: An HSM job is exported using the export_hsmtask command.

```
import_job_status -job_status_file <path>
```

path

Path to the Job Status container generated by FlashPro Express.

HSM flow only: imports and validates job status received from FlashPro Express or IHP. Status can be generated in the process of job execution to provide current job status, or as a result of job end, in which case, it includes cryptographically protected proof of job removal from the HSM.

CM HSM Programming Task

```
add_hsmtask_to_job -job_name <job>
                   -hsmtask_name <task>
                   [-m_request_type {INTERNAL|EXTERNAL}]
```

job

Name of the Programming Job for which task is created.

task

Name of the HSM Task. Must be unique within the Job.

m_request_type

Specifies how the M-HSM request is executed.

INTERNAL

This mode can only be specified if the same physical User HSM is used to generate and execute the programming job. In this case, the Job Manager will internally execute the request.

EXTERNAL

Default mode requires user to export the request, process it using FlashPro Express, and then import it back into the Job Manager project.

This command creates a new HSM Task for the specified Programming Job. The HSM task can contain one or more Job Tickets. This HSM task is then used to send a Job Request to the M-HSM and the Job Response received is imported into the HSM Task, which then enables HSM Task export.

```
new_hsmtask_ticket -job_name <job>
                   -hsmtask_name <task>
                   -ticket_name <ticket>
                   -device <device>
                   -actions <action>
                   -max_device <max>
```

job

Name of the Programming Job which is added a new ticket.

task

Name of the task within the Job.

ticket

Name of the new ticket. Must be unique within the task.

device

Name of the target device in the Job for the new ticket.

action

Programming action for the ticket.

max

Overbuild protection: max devices to use this ticket. Can be 'unlimited'.

This command creates a new Job Ticket for the HSM Task. The HSM task can have one or more Job Tickets for each device, but each Job Ticket is created per programming action. The overbuild protection parameter max_device is applicable to the protocols that are capable of controlling the number of authorized devices, such as the Authorization Code and Unique Key protocols.

```
hsmtask_m_request -job_name <job>
```

```
                -hsmtask_name <task>
                -request_file <path>
```

job

Name of the Programming Job that contains target HSM Task.

task

Name of the HSM Task for which CM Request is being generated.

path

Full file name of the request container.

This command creates a Job Request that is required to perform a handshake protocol with the M-HSM (FlashPro Express or IHP). Once FlashPro Express has processed the request, it generates and exports a Job Reply that must be imported into the HSM Task on the Job Manager side. This handshake protocol guarantees one time use of the HSM Task on the FlashPro Express or IHP side (M-HSM), thus preventing job replication. This command is only applicable if the HSM task was created with the request set to EXTERNAL value (default). See 'add_hsm_task' for details.

```
hsmtask_m_reply -job_name <job>
                -hsmtask_name <task>
                -reply_file <path>
```

job

Name of the Programming Job that owns target HSM Task.

task

Name of the task in the job.

path

Full file name to the container with Job Response.

Import Job Reply by FlashPro Express or IHP (M-HSM). This command is only applicable if the HSM task was created with the request set to EXTERNAL value (default). See 'add_hsm_task' for details. Refer to the 'process_job_request' FlashPro Express command documented in the *FlashPro Express User's Guide* .

```
export_hsmtask -job_name <job>
               -hsmtask_name <task>
               -location <path>
               -name <file_name>
```

job

Name of the Programming Job that contains task being exported.

task

Name of the task in the job.

path

Location of the export container.

file_name

File name for the export container.

This commands exports the HSM Job for further execution by FlashPro Express or IHP. This can only be executed after importing Job Reply.

SPI Directory

```
export_spi_directory -golden_ver <value or SPI file>
                     -golden_addr <hex value>
                     -update_ver <value or SPI file>
```

```
-update_addr <hex value>
-file <file_name>
```

-golden_ver <value or SPI file>

Specifies Golden SPI Image design version. There are two ways to specify the value:

- Decimal value less than 65536 (exclusive)
- SPI file from which the design version is read.

-golden_addr <hex value>

Specifies Golden SPI Image address where hex is 32-bit hexadecimal value with prefix 0x/0X.

-update_ver <value or SPI file>

Specifies Update SPI Image design version. There are two ways to specify the value:

- Decimal value less than 65536 (exclusive)
- SPI file from which the design version is read.

-update_addr <hex value>

Specifies Update SPI Image address where hex is a 32-bit hexadecimal value with prefix 0x/0X.

-file <file>

Mandatory argument; specifies the file export location.

Supported Families

SmartFusion2, IGLOO2

Examples

Both golden* options go together. The same is true for both update* options; the file argument is required:

```
export_spi_directory \
    -golden_ver {D:\flashpro_files\m2s025_jb_spi_dir\designer\a1_MSS\export\a1_MSS.spi} \
    -golden_addr {0xa} \
    -file {D:\flashpro_files\jobmgr_project12\dev.spidir}

export_spi_directory \
    -update_ver {456} \
```

A – Product Support

Microsemi SoC Products Group backs its products with various support services, including Customer Service, Customer Technical Support Center, a website, electronic mail, and worldwide sales offices. This appendix contains information about contacting Microsemi SoC Products Group and using these support services.

Customer Service

Contact Customer Service for non-technical product support, such as product pricing, product upgrades, update information, order status, and authorization.

From North America, call **800.262.1060**
From the rest of the world, call **650.318.4460**
Fax, from anywhere in the world, **650.318.8044**

Customer Technical Support Center

Microsemi SoC Products Group staffs its Customer Technical Support Center with highly skilled engineers who can help answer your hardware, software, and design questions about Microsemi SoC Products. The Customer Technical Support Center spends a great deal of time creating application notes, answers to common design cycle questions, documentation of known issues, and various FAQs. So, before you contact us, please visit our online resources. It is very likely we have already answered your questions.

Technical Support

For Microsemi SoC Products Support, visit http://www.microsemi.com/products/fpga-soc/design-support/fpga-soc-support.

Website

You can browse a variety of technical and non-technical information on the Microsemi SoC Products Group home page, at www.microsemi.com/soc.

Contacting the Customer Technical Support Center

Highly skilled engineers staff the Technical Support Center. The Technical Support Center can be contacted by email or through the Microsemi SoC Products Group website.

Email

You can communicate your technical questions to our email address and receive answers back by email, fax, or phone. Also, if you have design problems, you can email your design files to receive assistance. We constantly monitor the email account throughout the day. When sending your request to us, please be sure to include your full name, company name, and your contact information for efficient processing of your request.

The technical support email address is soc_tech@microsemi.com.

My Cases

Microsemi SoC Products Group customers may submit and track technical cases online by going to My Cases.

Outside the U.S.

Customers needing assistance outside the US time zones can either contact technical support via email (soc_tech@microsemi.com) or contact a local sales office.

Visit About Us for sales office listings and corporate contacts.

Sales office listings can be found at www.microsemi.com/soc/company/contact/default.aspx.

ITAR Technical Support

For technical support on RH and RT FPGAs that are regulated by International Traffic in Arms Regulations (ITAR), contact us via soc_tech_itar@microsemi.com. Alternatively, within My Cases, select **Yes** in the ITAR drop-down list. For a complete list of ITAR-regulated Microsemi FPGAs, visit the ITAR web page.

Microsemi Corporate Headquarters
One Enterprise, Aliso Viejo,
CA 92656 USA

Within the USA: +1 (800) 713-4113
Outside the USA: +1 (949) 380-6100
Sales: +1 (949) 380-6136
Fax: +1 (949) 215-4996

E-mail: sales.support@microsemi.com

About Microsemi

Microsemi Corporation (Nasdaq: MSCC) offers a comprehensive portfolio of semiconductor and system solutions for communications, defense & security, aerospace and industrial markets. Products include high-performance and radiation-hardened analog mixed-signal integrated circuits, FPGAs, SoCs and ASICs; power management products; timing and synchronization devices and precise time solutions, setting the world's standard for time; voice processing devices; RF solutions; discrete components; Enterprise Storage and Communication solutions, security technologies and scalable anti-tamper products; Ethernet solutions; Power-over-Ethernet ICs and midspans; as well as custom design capabilities and services. Microsemi is headquartered in Aliso Viejo, Calif. and has approximately 4,800 employees globally. Learn more at **www.microsemi.com**.

5-02-00709-4/05.18