Dell Australia Pty Limited, BSAFE Product Team

Dell BSAFE Entropy Source

Version 1.1

SP 800-90B Non-Proprietary
Public Use Document

Document Version: 1.4
Document Status: Released
Document Date: 10-Jul-2026

Prepared by:

TERON LABS

Level 2, 14 Moore St
Canberra, ACT 2601
Australia

+61 2 5114 4878
info@teronlabs.com
www.teronlabs.com

Vendor:

Dell Australia Pty Limited, BSAFE Product Team

Building A, 207 Pacific Highway
St Leonards NSW 2065
Australia

BSAFE
FIPS 140 Validated

DELL Technologies

Contents

Document Management 2
1. Description 3
2. Security Boundary 5
3. Operating Conditions 5
4. Configuration Settings 6
5. Physical Security Mechanisms 6
6. Conceptual Interfaces 7
7. Min-Entropy Rate 11
8. Health Tests 11
9. Maintenance 14
10. User Verification 14
11. Vendor Permissions and Relationship 14

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

Document Management

Version	Date	Author	Description
1.0	24-Mar-2026	Jasur Abdullaev, Yvonne Cliff	Initial Release
1.1	20-May-2026	Yvonne Cliff	Updated User Verification to list commands to retrieve version and name.
1.2	23-Jun 2026	Yvonne Cliff	Update Win 10 IoT OSR to 3 and remove ‘strength’ of 64.
1.3	26-Jun-2026	Yvonne Cliff	Minor wording updates
1.4	10-Jul-2026	Yvonne Cliff	Minor wording updates

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

1. Description

This document is the Public Use Document for the National Institute of Standards and Technology (NIST) SP 800-90B Entropy Source Validation of the Dell BSAFE Entropy Source version 1.1, which is a modification of the Jitter RNG 2.2.0 entropy source.[1]

This entropy source is a non-physical (NP) source and provides output in 256-bit blocks. The entropy source design is compliant with the ***shall*** statements of NIST SP 800-90B.

In a 'strength' parameter provided by the user to the entropy source, the user specifies the minimum number of bits of entropy each output block should provide. The validated strength is:

- 'strength' = **256** (providing **full entropy** output).

Although the entropy source operates correctly with any strength between 1 and 256, only the listed strength value is validated.

BSAFE Entropy Source (BES) version 1.1 is designed to be operated in Dell, Linux and Windows operating systems (e.g. Dell PowerProtect Data Domain, Dell PowerStoreOS, Dell OneFS, Red Hat Enterprise Linux, SUSE Linux Enterprise Server or Windows 10 IoT LTSC).

The evaluation was conducted using data and parameters measured in the evaluated version and configurations described within this document. The evaluated version of the entropy source / software is listed in Table 1 and the tested operational environments (OEs) are shown in Table 2.

The standard versions applicable to this submission are listed in Table 3.

Table 1 – Software and Entropy Source Versions

Type	Name	Description	Version
Entropy Source / Software	Dell BSAFE Entropy Source	The entropy source consists of software only.	1.1

[1] https://www.chronox.de/jent/

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

Table 2 – Tested Operational Environments

OE ID	Manufacturer	Model	Operational Environment & Version	Processor	OSR
Win 10 IoT	Dell	PowerEdge R760	Windows 10 IoT LTSC 2019 x86 (32-bit) on VMware ESXi 8	Intel Xeon Silver 4510 (Sapphire Rapids)	3
iDRAC 10	Dell	iDRAC10	Linux kernel 6 on Dell iDRAC10	Nuvoton NPCM8mnx (ARM Cortex A35)	1

Note: The entropy source does not make use of any Processor Algorithm Acceleration (PAA) function. As per NIST IG 2.3.C Additional Comment 6, testing with and without PAA/PAI was not required. In addition to the OEs listed in Table 2 without PAA, the entropy source is also certified on the same OEs with PAA.

Note: In accordance with Request for Guidance Response 241003, *Live Updates for Operating System*, each OE certification applies to versions of the operating system having the same major version.

Table 3 – Applicable Standards

Document	Title	Date
SP 800-90B	*Recommendation for Entropy Sources Used for Random Bit Generation*	Jan 2018
Implementation Guidance	*Implementation Guidance for FIPS 140-3 and the Cryptographic Module Validation Program* *IG 9.3.A last modified July 26, 2024* *IG D.J last modified November 5, 2021* *IG D.K last modified April 16, 2026* *IG D.O last modified May 4, 2021*	Apr 2026

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

2. Security Boundary

For all environments, the security boundary of the entropy source is the static library that contains its implementation. On Windows, this library is dellbes1.lib; on Unix-like platforms, it is libdellbes1.a.

The overall design of Dell BSAFE Entropy Source 1.1 entropy source and its core operations are illustrated in Figure 1. All items within the outer rectangle are included in the entropy source security boundary.

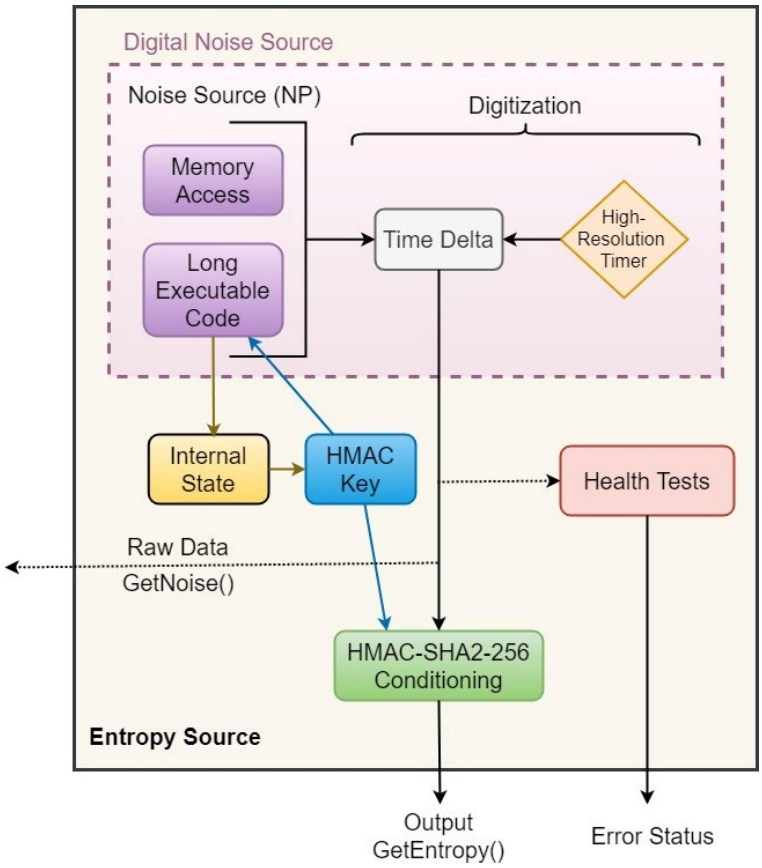


Figure 1: Entropy source design and security boundary

3. Operating Conditions

The entropy source is non-physical, and thus the operating conditions are inherited from the operational environments in which the entropy source is installed. The operating conditions for each of the OEs are shown in Table 4 below.

Table 4: Specifications of tested operational environments.

OE ID / Manufacturer / Model	Temperature (˚C)	Voltage (VAC)	Humidity (%)	Clock Speed (GHz)	Cache Sizes	OSR
Win 10 IoT	10 – 35	100 – 240	10 – 80	2.4 – 4.1	L1i: 12 x 32 KB L1d: 12 x 48 KB	3

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

OE ID / Manufacturer / Model	Temperature (˚C)	Voltage (VAC)	Humidity (%)	Clock Speed (GHz)	Cache Sizes	OSR
Dell PowerEdge R760 with Intel Xeon Silver 4510					L2: 12 x 2 MB L3: 30 MB	
iDRAC 10 Dell iDRAC 10	10 – 35	100 – 240	10 – 80	1.0	L1i: 4 x 32K L1d: 4 x 32K L2: 512K	1

4. Configuration Settings

The Dell BSAFE Entropy Source 1.1 configuration parameters are listed in Table 5.

Table 5: Dell BSAFE Entropy Source Configuration Parameters

Parameter	Description	Default Value	Validated Value
OSR	Oversampling rate (OSR) for the platform *Note: These values are the only validated OSR values.* *The default OSR can be used by passing an OSR value of 0 in the configuration when instantiating a new instance. If a non-zero OSR is provided in the configuration, it will override the default OSR.*	2 (Windows – must be overridden with a value of 3 upon instantiation by the FIPS module.) 1 (iDRAC)	3 (Windows) 1 (iDRAC)
strength (or the requested strength)	Value of the strength parameter passed to the entropy source when requesting entropy. *Note: The maximum value of the strength parameter is 256. Only the value listed is validated, although the entropy source will operate correctly for all values between 1 and 256 inclusive.*	None	256

5. Physical Security Mechanisms

The entropy source is non-physical, and thus the physical security mechanisms are inherited from the operational environments in which the entropy source is installed.

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

6. Conceptual Interfaces

The entropy source provides the following interfaces:

Initialization and Cleanup:

- BES_new

```
/**
 * Creates and initializes a @product object instance.
 *
 * @param [in]  config  The configuration for the new @product object.
 * @param [out] bes     The new @product object.
 *
 * @return #BES_OK indicates success.<br>
 *         For a list of return codes, see #BES_STATUS.
 */
BES_STATUS BES_CDECL BES_new(BES_CONFIG const *config, BES **bes);
```

- BES_delete

```
/**
 * Deletes a reference to a @product object.
 *
 * The reference count for the object is decremented and the pointer
 * is cleared. If there are no other references to the object, all
 * resources associated with the object are released.
 *
 * @param [in, out] bes  The @product object to delete a reference to.
 */
void BES_CDECL BES_delete(BES **bes);
```

- BES_CONFIG

  BES_CONFIG is a transparent read-only structure passing all the external configuration needed to create and use the entropy source.

  - Required configuration
    - Memory allocation callbacks (matching the BCM_CONFIG signature)
    - Oversampling rate (OSR) – as per Table 5, must be set to 3 for Win 10 IoT and 1 for iDRAC 10. iDRAC 10 may also use a value of 0 which defaults to an OSR of 1. This requirement is handled by the FIPS module.
  - Optional configuration
    - Logging callback (matching the BCM_CONFIG signature)
    - Personalization callback
      - This allows the calling module to provide some extra non-entropic data to initialize a unique entropy source
      - E.g. RDRAND output, process ID, CPU ID, /dev/urandom bytes, etc.

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

```
/**
 * Configuration for a new @product instance.
 *
 * @see BES_new()
 */
typedef struct
{
    /** The version of the #BES_CONFIG structure. */
    uint32_t version;

    /**
     * Optional user context for callbacks.
     *
     * This value is not used by @product, but is passed back unmodified
     * as the first argument of all user-defined callbacks.
     */
    void *user_ctx;

    /**
     * Optional over-sampling rate override.
     *
     * If non-zero, this value overrides the default built-in over-sampling
     * rate for the platform.
     * This value should be supplied as zero/default unless there is a known
     * need for changing the default over-sampling rate.
     */
    uint32_t osr;

    /**
     * User callback for memory allocation.
     *
     * @see BES_MEM_ALLOC_FUNC().
     */
    BES_MEM_ALLOC_FUNC *alloc_cb;

    /**
     * User callback for memory reallocation.
     *
     * @see BES_MEM_REALLOC_FUNC().
     */
    BES_MEM_REALLOC_FUNC *realloc_cb;

    /**
     * User callback for memory deallocation.
```


Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

```
     *
     * @see BES_MEM_FREE_FUNC().
     */
    BES_MEM_FREE_FUNC *free_cb;


    /**
     * Optional user callback for writing to a log file.
     *
     * If ``NULL``, no logging is performed for @product.
     *
     * @see BES_LOG_WRITE_FUNC().
     */
    BES_LOG_WRITE_FUNC *log_write_cb;


    /**
     * Optional user callback for providing entropy source
     * personalization data.
     *
     * If ``NULL``, no personalization data is applied.
     *
     * @see BES_PERSONALIZATION_FUNC().
     */
    BES_PERSONALIZATION_FUNC *personalization_cb;
}
BES_CONFIG;
```

- BES_ref_inc

```
/**
 * Increments the reference count for a @product object.
 *
 * @param [in] bes  The @product object to increment the reference count for.
 *
 * @return _bes_ if the reference increment is successful, otherwise `NULL`.
 */
BES * BES_CDECL BES_ref_inc(BES *bes);
```

Entropic Data Generation:

- BES_gather – this interface corresponds to the GetEntropy() conceptual interface from SP 800-90B, and obtains conditioned entropy for the caller.
    - It provides full entropy output for operational use when called with a strength of 256.

```
/**
 * Gather a number of bytes of entropy from @product.
```


Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

```
 *
 * @note @product gathers entropy in blocks of 256 bits (32 bytes).<br />
 *       The buffer length `len` must be a multiple of the block size.<br />
 *       Each output block will contain the requested `strength` in bits of
 *       entropy, spread across the block.
 *
 * @note Calling code must ensure this API is called from only a single
 *       thread at a time for a given #BES instance.
 *
 * @param [in] bes       The @product object to provide entropy
 * @param [in] strength  The strength in bits of the entropy to collect per
 *                       block of output.<br />
 *                       This must be between 1 and 256 bits.
 *                       Validated strength is 256 bits.
 * @param [in] buf       The buffer into which the bytes of entropy will be
 *                       placed.
 * @param [in] len       The length in bytes of `buf`. Must be a multiple of
 *                      32 bytes.
 *
 * @return #BES_OK indicates success.<br>
 *         For a list of return codes, see #BES_STATUS.
 */
BES_STATUS BES_CDECL BES_gather(BES *bes, unsigned int strength, void *buf,
size_t len);
```

- BES_deltas – this interface corresponds to the GetNoise() conceptual interface from SP 800-90B and can be used to gather noise source samples for entropy testing. It is a development-only function, not intended for use by FIPS modules or general applications. Deltas generated for this function are not used to generate entropy for the BES_gather function.

```
/*
 * Collect internal timestamp delta values from the entropy source.
 *
 * This function is not intended for use by FIPS modules or general
 * applications.
 * It is a development-only function to be used during testing.
 *
 * @param bes     [In]      The entropy source instance
 * @param count   [In]      The number of 64-bit delta values to collect
 * @param deltas  [In,Out]  The buffer to return the data in.<br />
 *                          Must be at least `count`*8 bytes in size.
 *
 * @return #BES_OK indicates success.<br>
 *         For a list of return codes, see @ref BES_STATUS.
 */
BES_STATUS BES_CDECL BES_deltas(BES *bes, size_t count, void *deltas);
```


Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

Status and Health Tests:

- BES_state – used to retrieve the state of the entropy source. Except for the stuck test, health test failures are permanent and require the entropy source instance to be discarded and a new instance generated via BES_delete and BES_new.

```
/**
 * Retrieves the state of a @product object.
 *
 * @param [in] bes  The @product object to query.
 *
 * @return #BES_STATE_READY indicates the entropy source is healthy and
 *         ready for use.<br>
 *         For a list of states, see #BES_STATE.
 */
BES_STATE BES_CDECL BES_state(BES *bes);
```

- On-demand health tests are performed by deallocating the current Jitter RNG instance and reallocating a new instance. The startup health tests are then re-executed and the samples used for those tests are discarded. This corresponds to the HealthTest conceptual interface from SP 800-90B.

7. Min-Entropy Rate

The entropy source compresses the requested strength plus 64 bits of entropy using HMAC-SHA2-256 to generate each 256-output block.

When the strength requested is 256, as required by this validation, the entropy source provides a min-entropy rate of 256 bits of min-entropy per 256-bit output block, or 8 bits per byte; i.e. full entropy output (based on FIPS 140-3 IG D.K Resolution 19).

8. Health Tests

The entropy source implements the following health tests:

- Stuck Test:
    - The test calculates whether the received time deltas are non-stuck, i.e. the first three discrete derivatives of the time stamp values are all non-zero.

  If this test fails, the entropy source will continue to gather more samples until enough non-stuck values are provided. No entropy is credited to stuck deltas.
- Repetition Count Test conforming to NIST SP 800-90B section 4.4.1:
    - Cutoff value, C = 22.

  The Repetition Count Test (RCT) is called to check each delta value when it is generated. The RCT operates on mapped data, namely the 8 least significant bits of the delta. These 8 bits are referred to as the sample.

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

The test compares the current sample value to the value of the previous sample. If they are the same, the count is incremented; otherwise, the count is reset to 0. If the count becomes greater than or equal to the cutoff, the BES_ERROR_HEALTH_TEST_FAILED error code is returned to the calling function.

- Adaptive Proportion Test conforming to NIST SP 800-90B section 4.4.2:
    - APT Window Size, W = 512;
    - Cutoff value, C = 311.

  The APT is called to check each delta value when it is generated. The APT operates on mapped data, namely the 8 least significant bits of the delta. These 8 bits are referred to as the sample.

  Failure of the APT results in the BES_ERROR_HEALTH_TEST_FAILED error code being returned to the calling function. This is the same error code as is used for the RCT failure, and the error is handled identically.

- Lag Predictor Test:
    - This test is a vendor-defined conditional test, and it is designed to detect a failure mode where the result becomes mostly deterministic.

  The LPT operates on unmapped data, namely the entire 64-bit delta.

  Failure of the LPT results in the BES_ERROR_HEALTH_TEST_FAILED error code being returned to the calling function. This is the same error code as is used for RCT and APT failure, and the error is handled identically.

In conformance with NIST SP 800-90B, these health tests are performed in two circumstances:

- Start-up Tests: As part of the creation of a new instance of the entropy source, the above health tests are run, in addition to a CAST on the HMAC implementation and a CAST on the stuck test.
- On-Demand Tests: On-demand health tests are performed by deallocating the current Jitter RNG instance and reallocating a new instance. The startup health tests are then re-executed, and the samples used for those tests are discarded.

The entropy source may return various error codes, described below.

- The entropy source only provides entropy to users in whole multiples of 256-bit blocks. It does not truncate blocks to provide output that is not a multiple of 256 bits in length. Requests for lengths which are not multiples of the 256-bit block size result in the BES_ERROR_BAD_LENGTH error code being returned to the calling function.
- Requests with a strength value greater than JITTER_MAX_STRENGTH also result in the BES_ERROR_BAD_LENGTH error code being returned to the calling function.
- Requests that result in a permanent health test failure return the BES_ERROR_HEALTH_TEST_FAILED error code.
- Other non-zero error codes may also be returned (e.g. due to internal error, memory allocation failure, etc.). Successful completion returns the BES_OK code.

The following codes are defined by the entropy source as possible return values. All error codes describe known failure modes except BES_OK.

Table 6: Dell BSAFE Entropy Source Configuration Parameters

Name	Value	Description

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

BES_OK	0	*The requested operation succeeded.*
BES_ERROR_FAILED	1	*The requested operation failed.*
BES_ERROR_ALLOC_FAILURE	2	*The requested operation failed to allocate memory.*
BES_ERROR_BUFFER_TOO_SMALL	3	*The requested operation detected an inadequately sized buffer.*
BES_ERROR_NULL_ARG	4	*A required argument was passed as NULL.*
BES_ERROR_BAD_VALUE	5	*Bad value supplied for a required argument.*
BES_ERROR_BAD_RANGE	6	*Value supplied for an argument was out of range.*
BES_ERROR_BAD_TYPE	8	*Value supplied for an argument was not a valid type.*
BES_ERROR_BAD_DATA	9	*Incorrect or invalid data supplied for an argument.*
BES_ERROR_BAD_LENGTH	10	*Invalid length supplied for an argument.*
BES_ERROR_NOT_FOUND	12	*A prerequisite was not found.*
BES_ERROR_NOT_SUPPORTED	13	*The requested operation is not supported.*
BES_ERROR_NOT_IMPLEMENTED	14	*The requested operation is not implemented.*
BES_ERROR_INVALID_STATE	15	*The object is not in the correct state for the requested operation.*
BES_ERROR_INTERNAL_ERROR	16	*An internal error occurred when performing an operation.*
BES_ERROR_BAD_CONFIG_VERSION	4000	*Unexpected #BES_CONFIG structure version.*
BES_ERROR_CONFIG_CALLBACK_UNAVAILABLE	4001	*Required configuration callback is unavailable.*
BES_ERROR_HEALTH_TEST_FAILED	4002	*Entropy source health test failed. The entropy source cannot be used.*

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.

BES_ERROR_SELF_TEST_CAST_FAILED	4003	*Cryptographic algorithm self-test failed. The entropy source cannot be used.*
_BES_ERROR_MAX	0x7fffffff	*UNDOCUMENTED*

During normal operation of the entropy source, when the RCT, APT or Lag Predictor test returns the BES_ERROR_HEALTH_TEST_FAILED error code, it will result in the following actions:

- The internal jitter->flags value is updated to indicate a health test failure.
- The delta collection is stopped and the BES_ERROR_HEALTH_TEST_FAILED error code is returned up the call stack.
- Previously generated 256-bit blocks of entropy remain in the buffer used to pass the entropy back to the calling application, but the current block and subsequent blocks in the buffer remain unchanged. However, no indication of the number of blocks written to the buffer is provided to the caller, apart from the error code which indicates to the caller that no entropy from the buffer should be used.
- The internal jitter->flags value remains set to an error value, and the instance of the entropy source will no longer produce any entropy. A new instance of the entropy source must be created to generate any additional entropy.

During the startup testing, a health test returning the error code will result in the following actions:

- The newly allocated entropy source is deleted.
- The error code from the health test is returned.

9. Maintenance

There are no maintenance requirements as this is a software-based entropy source.

10. User Verification

The entropy source performs the startup and continuous health tests described above, as required by SP 800-90B. No further testing by the user or module integrator is required or available.

The BES API provides functions that can be used to query the version and entropy source name at run-time. See **`bes.h: BES_version()`** and **`bes.h: BES_info()`**.

These values are also included directly in the **`bes_defs.h`** header file.

- BES_info(BES_INFO_FULLNAME) returns the string “Dell BSAFE Entropy Source 1.1.0.0”
- BES_version(&major, &minor, &patch) returns numeric major==1, minor==1, patch==0.

11. Vendor Permissions and Relationship

Reuse is restricted to the vendor.

Copyright © 2026 Dell Australia Pty Limited, BSAFE Product Team
This report shall not be reproduced except in full, including this copyright notice.