

JFES API Specification

Developer Notes

Public JavaScript API • Version 1.0

1. Purpose

This document specifies the public JavaScript API exposed by JFES. The API provides a deliberately small integration boundary between the JFES cryptographic engine and a calling application. Internal implementation state remains encapsulated; applications interact with JFES through the exported FES object.

2. Public API

```
const FES = {  
  fractalSBox: true,  
  setPasses: setPasses,  
  setSiloPath: setSiloPath,  
  encryptFES: encryptFES,  
  decryptFES: decryptFES  
};
```

3. API Summary

Member	Type	Purpose
fractalSBox	Boolean	Indicates that the Fractal S-Box transformation is enabled.
setPasses(value)	Function	Sets the number of FES transformation passes; valid values are 1 through 4.
setSiloPath(path)	Function	Selects and loads the Silo JSON used by JFES.
encryptFES(...)	Function	Encrypts a payload using the current JFES configuration.
decryptFES(...)	Function	Decrypts a payload using the current JFES configuration.

4. fractalSBox

The public configuration property indicates that the Fractal S-Box transformation is enabled. The current API initializes it to true.

Developer note: this property is exposed for configuration visibility; encryption and decryption should be performed through the public API functions.

5. setPasses(value)

Sets the number of transformation passes. The supplied value is converted to Number, validated as an integer from 1 through 4, and then stored in the private Locals state.

```
function setPasses(value) {  
  value = Number(value);  
  if (!Number.isInteger(value) || value < 1 || value > 4) {  
    throw new Error("FES passes must be an integer from 1 to 4.");  
  }  
}
```

```
    Locals.passes = value;
}
```

Default: 1. Valid range: 1–4 inclusive.

Developer note: passes is intentionally not exposed as a writable FES property. `setPasses()` is the enforcement boundary.

Example: `FES.setPasses(4);`

6. `setSiloPath(path)`

Selects and loads the Silo JSON at the supplied path. The selected Silo becomes the active Silo for subsequent FES operations.

Example: `await FES.setSiloPath("silo/Silo07.json");`

Developer note: callers must use `setSiloPath()` rather than accessing internal `siloPath` or `siloData` state.

7. Startup Silo Selection

An optional global `presetFESSiloPath` may be populated before JFES initialization. During `DOMContentLoaded` initialization, the preset is checked, the Silo is loaded when valid, and the preset is then cleared.

Example: `presetFESSiloPath = "silo/Silo07.json";`

Developer note: `presetFESSiloPath` is a startup integration mechanism. `setSiloPath()` is the public runtime interface.

8. Multi-Pass Processing

The fractal stream is generated to payload length multiplied by the configured pass count. Each pass consumes one payload-length segment of the continuous stream.

Encryption applies passes in forward order: Payload → Pass 1 → Pass 2 → Pass 3 → Pass 4 → Ciphertext.

Decryption applies the inverse transformations in reverse order: Ciphertext → Pass 4 inverse → Pass 3 inverse → Pass 2 inverse → Pass 1 inverse → Payload.

Developer note: applications do not need to reproduce this sequencing; `encryptFES()` and `decryptFES()` own the transformation process.

9. `encryptFES(...)`

Performs FES encryption using the active Silo, current pass count, and current JFES configuration. The exact argument and return signatures are those of the installed JFES build.

Developer note: do not access internal stream, portal, Silo, or transformation state directly.

FES ciphertext is always output in hex.

10. decryptFES(...)

Performs FES decryption using the active Silo and current pass count. Multi-pass decryption reverses the pass sequence and consumes the corresponding stream segment for each inverse transformation.

Ciphertext must always be in hex.

11. Typical Integration

```
FES.setPasses(4);
await FES.setSiloPath("silo/Silo07.json");

const ciphertext = await FES.encryptFES(...);
const plaintext  = await FES.decryptFES(...);
```

12. Encapsulation

Locals.passes, siloPath, siloData, portal navigation, fractal stream generation, and transformation internals are implementation details. The public FES object is the supported integration boundary.

13. Developer Notes

- Use FES.setPasses() to select 1–4 passes.
- Use FES.setSiloPath() to select the active Silo at runtime.
- Use presetFESSiloPath only when an application needs to select a Silo before initialization.
- Do not write internal Locals, siloPath, siloData, portal, or stream state from a consuming application.
- Applications should depend on the public FES API rather than the layout or names of internal implementation functions.
- The production artifact is minified and obfuscated; the public API remains the application integration boundary.

14. Complete Public API

```
const FES = {
  fractalSBox: true,
  setPasses: setPasses,
  setSiloPath: setSiloPath,
  encryptFES: encryptFES,
  decryptFES: decryptFES
};
```

This specification is based on the API surface and implementation details established for the current JFES build. Argument and return-value details for encryptFES() and decryptFES() are intentionally not fabricated where their exact signatures were not provided.

JFES — Public Cryptographic Artifact