Minutiae Interoperability Exchange III

Test Plan and Application Programming Interface

Last Updated: 28 February 2019

---

1 Overview

The Minutiae Interoperability Exchange III (MINEX III) test is an ongoing program to measure the performance of fingerprint matching software utilizing interoperable minutiae-based fingerprint templates. The content and format of those interoperable minutiae-based fingerprint templates are defined in this specification and are hereafter referred to as *MINEX III-compliant templates*.

Those wishing to submit software for MINEX III testing shall be required to provide the National Institute of Standards and Technology (NIST) with a software library implementing all functions of the Application Programming Interface (API) specified in Section 5. At a minimum, the software library submitted must provide functionality to create MINEX III-compliant templates based on individual fingerprint images. Support for matching pairs of MINEX III-compliant templates is strongly encouraged, but optional.

In addition to providing a general platform for testing the performance of interoperable fingerprint systems, MINEX III provides a mechanism for testing compliance with NIST Special Publication 800-76-2, Annex A.

2 What's New

If you previously participated in *Ongoing MINEX*, you'll find that the function signatures in the API remain unchanged. There are, however, a few technical changes you will need to be aware of before submittal.

- **Failures to match** might not necessarily return −1 (Section 5.4).
- **Threading** is not permitted (Section 6.1.1).
- A library **naming convention** is required (Section 6.1.3).
- The required **operating system** is now exclusively CentOS 7.6.1810 (Section 6.1.4).
- Submissions will be driven by a **validation** package (Section 6.3).
- **Timing** will be enforced and reported (Section 6.4).
- Greatly enhanced reporting enables new **client programs** (Section 7).
- Submissions won't be used if they fall out of **MINEX III Compliance** (Section 8).

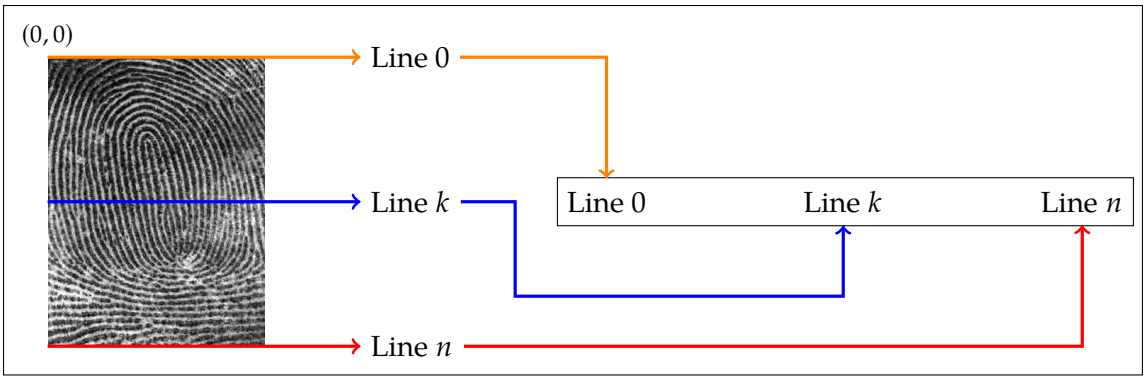

Figure 1: Order of scanned lines.

3 Fingerprint Image Data

3.1 Format

The software library must be capable of processing fingerprint images in uncompressed raw 8-bit (one byte per pixel) grayscale format. Images shall follow the scan sequence as defined by ISO/IEC 19794-4:2005, §6.2, and paraphrased here. Each image shall appear to have been captured in an upright position and approximately centered horizontally in the field of view. The image data shall appear to be the result of scanning a conventional inked impression of a fingerprint. Figure 1 illustrates the recording order for the scanned image. The origin is the upper left corner of the image. The X-coordinate (horizontal) position shall increase positively from the origin to the right side of the image. The Y-coordinate (vertical) position shall increase positively from the origin to the bottom of the image.

Raw 8-bit grayscale images are canonically encoded. The minimum value that will be assigned to a "black" pixel is zero. The maximum value that will be assigned to a "white" pixel is 255. Intermediate gray levels will have assigned values of 1–254. The pixels are stored left to right, top to bottom, with one 8-bit byte per pixel. The number of bytes in an image is equal to its height multiplied by its width as measured in pixels. The image height and width in pixels will be supplied to the software library as supplemental information.

3.2 Resolution and Dimensions

All images for this test will employ 500 pixels per inch resolution (horizontal and vertical).

The dimensions of the fingerprint images will vary from 150–812 pixels in width, and 166–1000 pixels in height. The software library must be capable of processing images with any dimensions within these specified ranges without the use of separately-invoked cropping or padding facilities. For example, software libraries that require cropping of large images must do so internal to the operation of the `create_template()` (Section 5.3).

3.3 Sensor and Impression Types

All images used for testing in MINEX III come from the POEBVA data set described in NISTIR 7296, Appendix B, Table 23, and thus have been obtained from live-scan sensors (Smiths-Heimann ACCO 1394 and Cross Match 300A). All images tested in MINEX III are plain impression type images.

4 NIST SP 800-76-2 Compliant Templates

To be considered MINEX III-compliant templates, all templates created must be compliant with NIST Special Publication 800-76-2, Annex A, Table 18.

Participants in MINEX04 may note that the requirements for templates specified by the MINEX III test are identical except for the fields listed below. These requirements are identical to those from Ongoing MINEX.

- In MINEX04, the field Finger Quality field had a range of values resulting from re-mapping the NIST NFIQ quality values (1 through 5) to the values 100, 75, 50, 25, and 1 respectively. However, NIST SP 800-76-2 re-maps these same NFIQ quality values to 100, 80, 60, 40, and 20 respectively. MINEX III uses NIST SP 800-76-2 Finger Quality values.
- In MINEX04, the field Impression Type had a range of 0–3. However, NIST SP 800-76-2 limits the range of values to 0 and 2. MINEX III uses NIST SP 800-76-2 Impression Type values.

5 Application Programming Interface

5.1 Predefined Constants

Constant		Explanation
0	MINEX_RET_SUCCESS	Success
1	MINEX_RET_BAD_IMAGE_SIZE	Image size not supported
2	MINEX_RET_FAILURE_UNSPECIFIED	Unspecified failure
3	MINEX_RET_FAILURE_BAD_IMPRESSION	Bad impression type
4	MINEX_RET_FAILURE_NULL_TEMPLATE	Null template used
5	MINEX_RET_FAILURE_BAD_VERIFICATION_TEMPLATE	Verification template error
6	MINEX_RET_FAILURE_BAD_ENROLLMENT_TEMPLATE	Enrollment template error

Table 1: MINEX III return codes.

Constant		Explanation
0	MINEX_FINGER_UNKNOWN	Unknown or unspecified position
1	MINEX_FINGER_RIGHT_THUMB	Right thumb
2	MINEX_FINGER_RIGHT_INDEX	Right index
3	MINEX_FINGER_RIGHT_MIDDLE	Right middle
4	MINEX_FINGER_RIGHT_RING	Right ring
5	MINEX_FINGER_RIGHT_LITTLE	Right little
6	MINEX_FINGER_LEFT_THUMB	Left thumb
7	MINEX_FINGER_LEFT_INDEX	Left index
8	MINEX_FINGER_LEFT_MIDDLE	Left middle
9	MINEX_FINGER_LEFT_RING	Left ring
10	MINEX_FINGER_LEFT_LITTLE	Left little

Table 2: MINEX III finger positions.

Constant		Explanation
0	MINEX_IMP_LIVESCAN_PLAIN	Live-scan, plain
2	MINEX_IMP_NONLIVESCAN_PLAIN	Non live-scan, plain

Table 3: MINEX III impression types.

Constant		Explanation
20	MINEX_QUALITY_POOR	Equivalent to NFIQ 5
40	MINEX_QUALITY_FAIR	Equivalent to NFIQ 4
60	MINEX_QUALITY_GOOD	Equivalent to NFIQ 3
80	MINEX_QUALITY_VERYGOOD	Equivalent to NFIQ 2
100	MINEX_QUALITY_EXCELLENT	Equivalent to NFIQ 1

Table 4: MINEX III quality values.

5.2 Identification

```
int32_t
get_pids(
    uint32_t *template_generator,
    uint32_t *template_matcher);
```

Figure 2: Obtain CBEFF PID information.

5.2.1 Parameters

- `template_generator` **(out)**: PID that identifies the participant's template generator.
- `template_matcher` **(out)**: PID that identifies the participant's template matcher.

5.2.2 Description

This function retrieves the CBEFF Product Identifier (PID) information that identifies the software library's core template generator and (if applicable) template matcher.

- The PID output for `template_generator` shall be identical in both format and value to the CBEFF Product defined by ANSI INCITS 378-2004, §6.4.4.
- If the software library does not support matching, the value in `template_matcher` shall be set to 0.
- Submissions will not be eligible for client program compliance if the CBEFF PIDs are not set.
- In the event of errors during the test, the version number portions of the CBEFF PID shall be incremented between different revisions of the software libraries sent to NIST.

5.2.3 Return Code

This method should return `MINEX_RET_SUCCESS` when successful. It is not expected to return anything other than `MINEX_RET_SUCCESS`.

5.3 Template Creation

```
int32_t
create_template(
    const uint8_t *raw_image,
    const uint8_t finger_quality,
    const uint8_t finger_position,
    const uint8_t impression_type,
    const uint16_t height,
    const uint16_t width,
    uint8_t *output_template);
```

Figure 3: Create a MINEX III compliant template from a raw image.

5.3.1 Parameters

- `raw_image` **(in)**: The uncompressed image data.
- `finger_quality` **(in)**: Quality of fingerprint depicted in `raw_image` (Table 4).
- `finger_position` **(in)**: Position of the fingerprint depicted in `raw_image` (Table 2).
- `impression_type` **(in)**: Impression of the finger depicted in `raw_image` (Table 3).
- `height` **(in)**: Number of rows in `raw_image`.
- `width` **(in)**: Number of columns in `raw_image`.
- `output_template` **(in, out)**: A pre-allocated memory location where the template shall be written.

5.3.2 Description

This function takes a raw image as input and outputs a corresponding MINEX III-compliant template.

- Memory for `output_template` will have been allocated by NIST before the call. Implementations of `create_template()` shall **not** allocate `output_template`, but should simply write to it starting at offset 0.
- The software library shall always write to `output_template`, even if the software library cannot extract features. In failure conditions, a 32-byte template (a 26-byte ANSI INCITS 378-2004 record header, a 4-byte finger view header, and a 2-byte extended data block length) with zero minutiae might be created.

5.3.3 Return Code

This method should return `MINEX_RET_SUCCESS` when successful, or another approved return code on failure. Regardless of failure, `output_template` shall be set to a valid MINEX III-compliant template. All templates will be used during matching, even templates with zero minutiae.

5.4 Template Matching

```
int32_t
match_templates(
    const uint8_t *verification_template,
    const uint8_t *enrollment_template,
    float *similarity);
```

Figure 4: Compare two MINEX III-compliant templates.

5.4.1 Parameters

- `verification_template` **(in)**: A template returned from `create_template()`, though not necessarily this software library's implementation.
- `enrollment_template` **(in)**: A template returned from `create_template()`, though not necessarily this software library's implementation.
- `similarity` **(out)**: A score representing the similarity of the original fingerprint images represented by templates `verification_template` and `enrollment_template`.

5.4.2 Description

This functions compares two MINEX III-compliant templates and outputs a value indicating their similarity. `enrollment_template` shall be compared to the `verification_template`, in that precise order (in the event that the underlying matcher is order-dependent). The similarity score returned is a floating-point number that represents the *similarity* of the original fingerprint images from which the templates were created. Scores should **not** be quantized.

- Memory for `similarity` will be allocated by NIST.
- `similarity` shall always be modified, even in the event of a failure.
- NIST Special Publication 800-76-2, Annex A, §3.2 does not permit the match operation to fail. A failure or refusal to compare the inputs shall in all cases result in the modification of `similarity`, ideally setting `similarity` to a low value.
- Minutiae quality values in templates will always be set to 0 when passed to this function, even if `create_template` set the values.
- This method should set `similarity` to -1 if a template matcher is not implemented (template generator submission only).

5.4.3 Return Code

This method should return `MINEX_RET_SUCCESS` when successful, or another approved return code on failure. Regardless of failure, `similarity` shall be modified, and all `similarity` will be used when calculating false non-match rate (FNMR) regardless of return code.

6 Software and Documentation

6.1 Software Library and Platform Requirements

The functions specified in Section 5 shall be implemented exactly as defined in a software library. The header file used in the MINEX III test driver is provided on the MINEX III website in the MINEX III validation package. The symbols for these functions shall be exported as C symbols for the backwards compatibility of previous tests in the MINEX family.

6.1.1 Restrictions

Individual software libraries provided must not include multiple modes of operation or algorithm variations. No switches or options will be tolerated within one library. For example, the use of two different encoding algorithms by a minutiae extractor would be split across two separate software libraries (though the MINEX III application indicates that NIST will only accept one submission every 90 days).

Participants shall provide NIST with binary code in the form of a software library only (i.e., no source code or headers). It is preferred that the software libraries be submitted in the form of a *single* dynamic/shared library file (i.e., ".so" file), however static libraries (i.e., ".a" files) are permitted if technically required.

The software library shall not make use of threading. The NIST test driver operates as a Message Passing Interface (MPI) job over multiple compute nodes, and then forks itself into many processes. In the test environment, there is no advantage to threading. It limits the usefulness of NIST's batch processing and makes it impossible to compare timing statistics across MINEX III participants.

6.1.2 External Dependencies

It is preferred that the API specified by this document be implemented in a single "core" library. Additional libraries may be submitted that support this "core" library file (i.e., the "core" library file may have dependencies implemented in other libraries if a single library is not feasible).

Note that dependencies on external software libraries such as compiler-specific development environment libraries are discouraged. If absolutely necessary, external libraries must be provided to NIST upon prior approval by the MINEX III Liaison.

Use of processor optimizations is allowed and encouraged, but software libraries must be able to run on the following processor types:

- AMD Opteron 8376HE
- Intel Xeon E5405, X5680, X5690, X7560
- Intel Xeon E5-2680

6.1.3 Naming

The "core" software library submitted for MINEX III shall be named in a predefined format. The first part of the software library's name shall be `libminexiii_`. The second piece of the software library's name shall be a non-infringing unique identifier that matches the regular expression `[A-Za-z0-9]+` (likely your organization's name), followed by an underscore. The final part of the software library's name shall be a version number, followed by a file extension. Supplemental libraries may have any name, but the "core" library must be dependent on supplemental libraries in order to be linked correctly. The **only** library that will be explicitly linked to the MINEX III test driver is the "core" library, as demonstrated in Section 6.1.4.

In the event that the software library is being submitted for client program compliance, the version number shall match the CBEFF PID version number of the template generator, as returned by `get_pids()`. Otherwise, the number shall be an integer, starting with 1, and shall be incremented by 1 for all subsequent submissions, including later unrelated submissions by the same organization. With this naming scheme, **every "core" library received by NIST shall have a unique filename**. Incorrectly named or versioned software libraries will be rejected.

Example
Initech submits a MINEX III shared library named `libminexiii_initech_0001.so` with version 1.0 of their algorithm for client program certification. This library assigns `0x12340001` to the parameter `template_generator` in `get_pids()`. NIST determines that Initech's validation fails and rejects the library. Initech submits version 1.0.1 to fix the bug in 1.0. The new name of their library is `libminexiii_initech_0002.so` and they updated the CBEFF PID in their implementation of `get_pids()` to `0x12340002`. For anonymity to evaluation administrators during testing, Initech may have elected to name their library `libminexiii_1234_0002.so`, using the CBEFF Product ID as their unique identifier.

6.1.4 Testing Procedure

The software library will be tested in non-interactive "batch" mode (i.e., without terminal support) in an isolated environment (i.e., no Internet connectivity). Thus, the software library shall not use any interactive functions, such as graphical user interface calls, or any other calls that require terminal interaction (e.g., writes to `stdout`).

NIST will link the provided library files to a C++ language test driver application using the compiler `g++` (version `4.8.5 20150623 (Red Hat 4.8.5-36)`) under **CentOS 7.6.1810**. For example:

```
g++ -o minexiii minexiii.cpp -L. -lminexiii_initech_0001
```

Participants are required to provide their software libraries in a format that is linkable using `g++` with the NIST test driver. All compilation and testing will be performed on 64-bit hardware running CentOS 7.6.1810. Thus, participants are strongly encouraged to verify library-level compatibility with `g++` on CentOS 7.6.1810 **prior to** submitting their software to NIST to avoid unexpected problems.

6.2 Usage

The software library shall be executable on any number of machines without requiring additional machine-specific license control procedures, activation, hardware dongles, or any other form of rights management.

The software library usage shall be **unlimited**. No usage controls or limits based on licenses, execution date/time, number of executions, etc., shall be enforced by the software library. Should a limitation be encountered, the software library shall have all client program certifications revoked.

6.3 Validation and Submitting

NIST shall provide a *validation package* that will link the participant "core" software library to a sample test driver. Once the validation successfully completes on the participant's system, a file with validation data and the participant's software library will be created. After being encrypted, **only this** file and a public key shall be submitted to NIST. Any software library submissions not generated by an unmodified copy of NIST's MINEX III validation package will be rejected. Any software library submissions that generate errors while running the validation package on either the participant's or NIST's hardware will be rejected. Any software library submissions not generated with the current version of NIST's MINEX III validation package will be rejected. Any submissions of successful validation runs not created on CentOS 7.6.1810 will be rejected.

6.4 Speed

A template match operation shall take no more than 10 milliseconds on average to complete. A template creation operation shall take no more than 500 milliseconds on average to complete. Timing tests will be run, enforced, and reported on a sample of the MINEX III dataset prior to completing the entire test. These speeds are based on the current processor being used, which is the **Intel Xeon E5-2680**, and are based on the values present in NIST Special Publication 800-76-2, §4.5.2.

7 Client Programs

The MINEX family of tests were created to serve as a benchmark for the biometric specifications of FIPS 201 as a part of the U.S. Government's PIV program. Over the years, the test's results have been used by others for a multitude of different initiatives. Adding enhanced reporting on these algorithms will make it easier to apply the results of MINEX III to organizations and programs requiring a compliance threshold for interoperable fingerprint template generators and matchers.

To facilitate this, MINEX III will introduce the concept of "client programs." A client program refers to a biometric program defining a set of guidelines that encompass compliance, with the U.S. Government's PIV program serving as just one example. If other well-defined client programs whose operational protocol is in line with MINEX III are encountered, adherence to that program may be reported. Please note that MINEX III will remain unaffiliated with any organization

other than the U.S. Government, and posted results should **not be considered official**, *except* for PIV.

8 MINEX III Compliance

In an effort to reduce complexity and modernize the test, participation guidelines have been enacted in MINEX III which will form "MINEX III Compliance." A MINEX III submission must meet these criteria in order to be used during the interoperable test. Once a submission fails to adhere to one or more of the guidelines, it will be removed from the test. Compliance requirements for client programs are maintained separately. NIST may update these guidelines from time to time. A visual representation is available in Figure 5.

MINEX III compliance shall be determined by:

1. Functionality: the submission passes MINEX III validation, which includes limits on algorithm speed, memory consumption, runtime, etc.,
   (a) An exemption is made for template generators that successfully passed the Ongoing MINEX validation prior to the start of MINEX III.
2. Interoperability: all MINEX III-compliant template matchers match templates from the submitted template generator with a FNMR $\leq 10^{-2}$ at FMR $\leq 10^{-2}$ (PIV Level 1),
3. Interoperability: a template matcher is submitted and it matches templates from all MINEX III-compliant template generators with a FNMR $\leq 10^{-2}$ at FMR $\leq 10^{-2}$ using two fingers (PIV Level 1),
4. Accuracy at operationally-typical FMR: a template matcher is submitted and it matches its corresponding template generator's templates with a FNMR $\leq (2\times10^{-2})$ at FMR $\leq 10^{-4}$ using one finger (PIV Level 2),
   (a) An exemption is made for PIV-compliant template generators from Ongoing MINEX.
5. Compliance with minutia placement standards: minutiae density plots derived from generated templates do not exhibit a periodic, grid-like, or geometric structure without reasonable justification,
6. History: the validation package was submitted within the last 5 years,
   (a) An exemption is made for PIV-compliant template generators from Ongoing MINEX.
7. Diversity of operating points: a template matcher is submitted and produces at least 512 unique scores over the entire dataset when comparing fingerprint templates from two different subjects,
8. Availability: the submission is one of two submissions from the submitting organization, or its subsidiaries, acquisitions, or mergers, by first replacing submissions with the same CBEFF Product IDs and older version numbers, followed by replacing submissions with older acceptance dates.
   (a) A deviation is made for submissions exempted by 1a, 4a, or 6a. Once *any* MINEX III submission is received by an organization or the organization's subsidiaries, acquisitions,

or mergers, *all* exempted submissions from the submitting organization, and the organization's subsidiaries, acquisitions, and mergers will lose their exemption, regardless of the submission's compliance status.

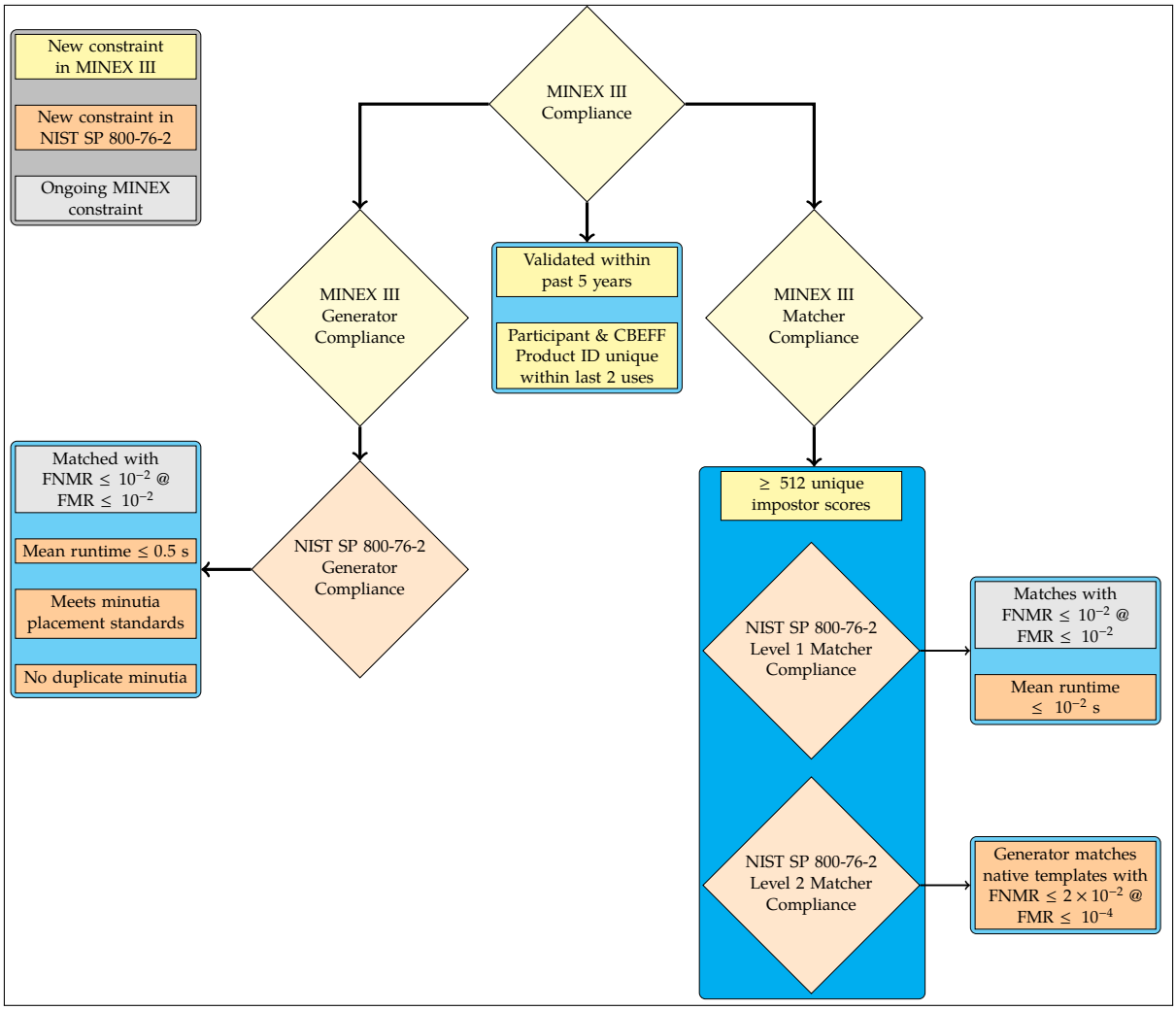

Figure 5: Visual representation of the MINEX III compliance constraints defined in Section 8.

Disclaimer

Certain commercial equipment, instruments, or materials are identified in this paper in order to specify the experimental procedure adequately. Such identification is not intended to imply recommendation or endorsement by the National Institute of Standards and Technology, nor is it intended to imply that the materials or equipment identified are necessarily the best available for the purpose.

Revision History

28 February 2019 Changed operating system from CentOS 7.2.1511 to CentOS 7.6.1810.

13 July 2016 Changed operating system from CentOS 7.0.1406 to CentOS 7.2.1511.

12 May 2016 Requirement for number of unique impostor scores increased from 256 to 512.

07 July 2015 Processor used for timing was updated.

11 June 2015 Initial revision.