National Software Reference Library Reference Dataset Version 3

February 7, 2023

The National Software Reference Library (NSRL) is beginning the process of transitioning from the current Reference Data Set (RDS) format, RDSv2, to an updated format, to be known as RDSv3. The new RDSv3 format will include a number of major changes, which include publication as an SQLite3 database, the inclusion of SHA-256 hashes for published files, and incremental releases, along with other changes.

As a SQLite database publication, the RDSv3 format will allow users to more easily manipulate the data published in the RDS, while also including more product and file metadata. The RDSv3 publication also includes a more modern set of hashes, with the publication of SHA-256 file hashes, and the removal of CRC-32 file hashes. A copy of the current SQLite database schema for the RDSv3 publication is included at the end of this document, along with being included in the RDS_2021.12.2_curated.zip file published on the NSRL website.

With the RDSv3 publication, it will also be possible for users to construct an RDSv2 like publication from data included in the RDSv3 format. Data in the four core NSRL files in RDSv2, NSRLFile.txt, NSRLMfg.txt, NSRLOS.txt, and NSRLProd.txt, will be stored in a set of four VIEWS in the RDSv3 SQLite publication, known as FILE, MFG, OS, and PKG (these views are defined bellow in the included RDSv3 database schema). The NSRL plans to provide a method for users to convert an RDSv3 publication into an RDSv2 like publication, for those who are interested.

RDSv3 will follow the same quarterly publication schedule currently used for RDSv2; the NSRL will publish RDS sets no later than the first Friday in the months of March, June, September, and December. There are two formats in which the NSRL will release RDSv3 publications, 1) a complete and full publication of the four RDS sets: Modern, Legacy, Android, and iOS, and 2) a "delta" publication of the four RDS sets: Modern, Legacy, Android, and iOS, covering changes that have occurred in the NSRL database since the previous RDSv3 publication. It is planned that the NSRL will be publishing complete and full RDSv3 publications in the first quarter of each year (March), and publish delta RDSv3 publications that build on the full publication in all four quarters (March, June, September, December). The delta RDSv3 publications will be a set of SQL INSERT, UPDATE, and DELETE statements contained in a single SQL file, that can be run in the previous full RDSv3 SQLite database publication. The NSRL will provide instructions for constructing the most up to date full RDSv3 publication from a past release and the latest delta set publication.

The RDSv3 downloads will be available on the NSRL website's download page as a zip file. The contents of the zip file will depend on whether the publication is a full database release, or a delta release. In the full database publication, the zip file will contain the following:

- RDS_YYYY.MM.<patch>_<set>.db

    - (ex. RDS_2022.03.1_modern.db)
    - SQLite database publication of the RDS set
- RDS_YYYY.MM.<patch>_<set>.schema.sql
    - (ex. RDS_2022.03.1.schema.sql)
    - Current schema for the SQLite database publication
- readme.txt
    - Readme file containing information about the publication
- signatures.txt
    - SHA-1 hash signatures of the three above files published in the RDS set

In the delta RDSv3 publication, the zip file will contain the following:

- RDS_YYYY.MM.<patch>_<set>_delta.sql
    - Full set of SQL INSERT, UPDATE, and DELETE statements needed in order to update the previous RDSv3 database file to the latest version of published data.
    - It is intended that the user open the previous RDS database version in SQLite, and then run all statements in the given file. This can be done by using the following command while connected to the previous release database:
        - .read RDS_YYYY.MM.<patch>_<set>_delta.sql
        - Ex) .read RDS_2022.01.3_curated_delta.sql
    - This process will take a few minutes to complete.
    - Further update instructions can be found in this document below.
- RDS_YYYY.MM.<patch>_<set>.schema.sql (ex. RDS_2022.01.3.schema.sql)
    - Current schema for the SQLite database publication
- readme.txt
    - Readme file containing information about the delta publication
- signatures.txt
    - SHA-1 hash signatures of the above files to be published in the RDS set

For all questions, comments, or feedback regarding the transition to the RDSv3 publication, please contact us at nsrl@nist.gov .

For current information about the NSRL and RDS, please see https://www.nsrl.nist.gov or contact the project team at nsrl@nist.gov .

Updating an RDSv3 SQLite Database Using a Delta Publication

It is intended that updating a full database with a delta publication be done on an un-altered version of the full database publication of the same publication set. The delta publication should be run on the most recent previous full database publication of the same set.

Updating Via SQLite GUI Tool

On all operating systems, you must first move the downloaded delta SQL file to the same location as your RDSv3 database file. This includes adding the sha file containing the expected sha1 value of the RDSv3 database after updating via the delta publication to the same directory.

There are a number of SQLite GUI tools that can be used to interact with the RDSv3 SQLite database. These tools can work on any Windows, Mac, or Linux operating systems. Open the tool and the RDSv3 SQLite database from within the tool. Once in the tool, a previous version of the RDSv3 database can be updated, using the delta SQL file from within these tools anywhere you can enter and run direct queries.

While in a SQLite GUI tool, run the following command.

```
.read RDS_2022.01.3_curated_delta.sql
```

Do NOT copy and paste the entire contents of the delta SQL file directly into a SQLite GUI tool, as the delta SQL file will likely contain millions of lines.

Once updated, it is recommended that you close the database and GUI tool, in order to perform checks on the database. This will require opening a command line. See below the steps for validating the updated database with the provided sha file in the delta publication, using the command line on your preferred operating system.

Updating Via Command Line

On all operating systems, you must first move the downloaded delta SQL file to the same location as your RDSv3 database file. This includes adding the sha file containing the expected sha1 value of the RDSv3 database after updating via the delta publication to the same directory.

Windows

Install sqlite3

Updating the RDSv3 database

```
cd to the directory where your RDSv3 database is stored
sqlite3
sqlite3 RDS_2021.12.2_curated.db
.read RDS_2022.01.3_curated_delta.sql
.q
ren RDS_2021.12.2_curated.db RDS_2022.01.3_curated.db
```

Mac OS and Linux

Install sqlite3

Updating the RDSv3 database

```
cd to the directory where your RDSv3 database is stored
sqlite3 RDS_2021.12.2_curated.db
.read RDS_2022.01.3_curated_delta.sql
.q
mv RDS_2021.12.2_curated.db RDS_2022.01.3_curated.db
```

RDSv3 SQLite Database Publication Schema

The following RDSv3 SQLite database publication schema is also available for download on the RDSv3 demonstration set published on the NSRL website.

```
CREATE TABLE PACKAGE_OBJECT (
        package_object_id       INTEGER UNIQUE  NOT NULL,
        package_id      INTEGER         NOT NULL,
        object_id               INTEGER UNIQUE  NOT NULL,
        CONSTRAINT PK_PACKAGE_OBJECT__PACKAGE_OBJECT_ID PRIMARY KEY (package_object_id)
);
CREATE TABLE APPLICATION (
        application_id  INTEGER         UNIQUE          NOT NULL,
        package_id      INTEGER                         NOT NULL,
        name                VARCHAR             DEFAULT ''      NOT NULL,
        name_b64            VARCHAR                             NOT NULL,
        name_coding         VARCHAR                             NOT NULL,
        version             VARCHAR             DEFAULT '',
        poe                 VARCHAR             DEFAULT 'purchased'     NOT NULL,
        build               VARCHAR             DEFAULT '',
        latest_copyright    VARCHAR             DEFAULT '',
        other               VARCHAR             DEFAULT '',
        creation_date   TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        CONSTRAINT PK_APPPLICATION__APPLICATION_ID PRIMARY KEY (application_id),
        CONSTRAINT FK_APPLICATION__PACKAGE_ID FOREIGN KEY (package_id) REFERENCES PACKAGE_OBJECT
(package_id)
);
CREATE TABLE APPLICATION_TYPE (
        application_type_id     INTEGER UNIQUE                  NOT NULL,
        description     VARCHAR                         NOT NULL,
        creation_date   TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        CONSTRAINT PK_APPLICATION_TYPE__APPLICATION_TYPE_ID PRIMARY KEY (application_type_id)
);
CREATE TABLE APPLICATION_APPLICATION_TYPE (
        application_application_type_id  INTEGER UNIQUE  NOT NULL,
        application_id                  INTEGER         NOT NULL,
        application_type_id                     INTEGER         NOT NULL,
        CONSTRAINT PK_APP_APP_TYPE__APPLICATION_APPLICATION_TYPE_ID PRIMARY KEY
(application_application_type_id),
        CONSTRAINT FK_APPLICATION_APPLICATION_TYPE__APPLICATION_ID FOREIGN KEY (application_id) REFERENCES
APPLICATION (application_id),
        CONSTRAINT FK_APPLICATION_APPLICATION_TYPE__APPLICATION_TYPE_ID FOREIGN KEY (application_type_id)
REFERENCES APPLICATION_TYPE (application_type_id)
);
CREATE TABLE LANGUAGE (
        language_id     INTEGER UNIQUE                  NOT NULL,
        name            VARCHAR DEFAULT ''              NOT NULL,
        creation_date   TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        language_tag    VARCHAR,
        CONSTRAINT PK_LANGUAGE_LANGUAGE_ID PRIMARY KEY (language_id)
);
CREATE TABLE APPLICATION_LANGUAGE (
        application_language_id  INTEGER UNIQUE  NOT NULL,
        language_id             INTEGER         NOT NULL,
        application_id          INTEGER         NOT NULL,
        CONSTRAINT PK_APPLICATION_LANGUAGE__APPLICATION_LANGUAGE_ID PRIMARY KEY (application_language_id),
        CONSTRAINT FK_APPLICATIONI_LANGUAGE__APPLICATION_ID FOREIGN KEY (application_id) REFERENCES
APPLICATION (application_id),
        CONSTRAINT FK_APPLICATION_LANGUAGE__LANGUAGE_ID FOREIGN KEY (language_id) REFERENCES LANGUAGE
(language_id)
);
CREATE TABLE OPERATING_SYSTEM (
        operating_system_id     INTEGER UNIQUE                  NOT NULL,
```

```
        name            VARCHAR DEFAULT ''              NOT NULL,
        name_b64        VARCHAR,
        name_coding     VARCHAR,
        version         VARCHAR DEFAULT ''              NOT NULL,
        architecture    VARCHAR DEFAULT ''              NOT NULL,
        creation_date   TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        CONSTRAINT PK_OPERATING_SYSTEM__OPERATING_SYSTEM_ID PRIMARY KEY (operating_system_id)
);
CREATE TABLE OPERATING_SYSTEM_APPLICATION (
        operating_system_application_id  INTEGER UNIQUE  NOT NULL,
        operating_system_id                     INTEGER         NOT NULL,
        application_id                  INTEGER         NOT NULL,
        CONSTRAINT PK_OPERATING_SYSTEM_APP__OPERATING_SYSTEM_APP_ID PRIMARY KEY
(operating_system_application_id),
        CONSTRAINT FK_OPERATING_SYSTEM_APPLICATION__OS_APPLICATION_ID FOREIGN KEY (operating_system_id)
REFERENCES OPERATING_SYSTEM (operating_system_id),
        CONSTRAINT FK_PLATFORM_APPLICATION__APPLICATION_ID FOREIGN KEY (application_id) REFERENCES
APPLICATION (application_id)
);
CREATE TABLE MANUFACTURER (
        manufacturer_id INTEGER UNIQUE                  NOT NULL,
        name            VARCHAR DEFAULT ''              NOT NULL,
        name_b64        VARCHAR,
        name_coding     VARCHAR,
        address1        VARCHAR DEFAULT ''              NOT NULL,
        address1_b64    VARCHAR,
        address1_coding VARCHAR,
        address2        VARCHAR DEFAULT ''              NOT NULL,
        address2_b64    VARCHAR,
        address2_coding VARCHAR,
        city            VARCHAR DEFAULT ''              NOT NULL,
        city_b64        VARCHAR,
        city_coding     VARCHAR,
        stateprov               VARCHAR DEFAULT ''              NOT NULL,
        postal_code     VARCHAR DEFAULT ''              NOT NULL,
        country         VARCHAR DEFAULT ''              NOT NULL,
        telephone               VARCHAR DEFAULT ''              NOT NULL,
        fax             VARCHAR DEFAULT ''              NOT NULL,
        url             VARCHAR DEFAULT ''              NOT NULL,
        url_b64         VARCHAR,
        url_coding      VARCHAR,
        email           VARCHAR DEFAULT ''              NOT NULL,
        creation_date   TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        CONSTRAINT PK_MANUFACTURER_MANUFACTURER_ID PRIMARY KEY (manufacturer_id)
);
CREATE TABLE MANUFACTURER_APPLICATION (
        manufacturer_application_id     INTEGER UNIQUE  NOT NULL,
        manufacturer_id         INTEGER         NOT NULL,
        application_id          INTEGER         NOT NULL,
        CONSTRAINT PK_MANUFACTURER_APPLICATION__MANUFACTURER_APPLICATION_ID PRIMARY KEY
(manufacturer_application_id),
        CONSTRAINT FK_MANUFACTURER_APPLICATION__APPLICATION_ID FOREIGN KEY (application_id) REFERENCES
APPLICATION (application_id),
        CONSTRAINT FK_MANUFACTURER_APPLICATION__MANUFACTURER_ID FOREIGN KEY (manufacturer_id) REFERENCES
MANUFACTURER (manufacturer_id)
);
CREATE TABLE MANUFACTURER_OPERATING_SYSTEM (
        manufacturer_operating_system_id INTEGER UNIQUE  NOT NULL,
        operating_system_id                     INTEGER         NOT NULL,
        manufacturer_id                 INTEGER         NOT NULL,
        CONSTRAINT PK_MANUFACTURER_OPERATING_SYSTEM__MANUFACTURER_OS_ID PRIMARY KEY
(manufacturer_operating_system_id),
        CONSTRAINT FK_MANUFACTURER_OPERATING_SYSTEM__MANUFACTURER_ID FOREIGN KEY (manufacturer_id)
REFERENCES MANUFACTURER (manufacturer_id),
        CONSTRAINT FK_MANUFACTURER_OPERATING_SYSTEM__OPERATING_SYSTEM_ID FOREIGN KEY (operating_system_id)
REFERENCES OPERATING_SYSTEM (operating_system_id)
);
```

```
CREATE TABLE METADATA (
        metadata_id     NUMERIC UNIQUE                  NOT NULL,
        object_id               INTEGER,
        key_hash        VARCHAR                         NOT NULL,
        image_hash      VARCHAR,
        path            VARCHAR                         NOT NULL,
        path_b64        VARCHAR,
        path_coding     VARCHAR,
        file_name               VARCHAR                         NOT NULL,
        file_name_b64   VARCHAR,
        file_name_coding VARCHAR,
        extension               VARCHAR DEFAULT ''              NOT NULL,
        extension_b64   VARCHAR,
        extension_coding VARCHAR,
        bytes           INTEGER                         NOT NULL,
        mtime           TIMESTAMP,
        used_in_rds     TIMESTAMP,
        update_date     TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        recursion_level INTEGER,
        extractee_id    INTEGER DEFAULT 0               ,
        crc32   VARCHAR NOT NULL,
        md5             VARCHAR NOT NULL,
        sha1    VARCHAR NOT NULL,
        sha256  VARCHAR NOT NULL,
        CONSTRAINT PK_METADATA__METADATA_ID PRIMARY KEY (metadata_id),
        CONSTRAINT FK_METADATA__EXTRACTEE_ID FOREIGN KEY (extractee_id) REFERENCES METADATA (metadata_id),
        CONSTRAINT FK_METADATA__OBJECT_ID FOREIGN KEY (object_id) REFERENCES PACKAGE_OBJECT (object_id)
);
CREATE TABLE VERSION (
        version VARCHAR UNIQUE  NOT NULL,
        build_set       VARCHAR NOT NULL,
        build_date      TIMESTAMP       DEFAULT CURRENT_TIMESTAMP       NOT NULL,
        release_date    TIMESTAMP       NOT NULL,
        description     VARCHAR NOT NULL,
        CONSTRAINT PK_VERSION__VERSION PRIMARY KEY (version)
);
CREATE VIEW FILE AS
    SELECT
        UPPER(md.sha256) AS sha256,
        UPPER(md.sha1) AS sha1,
        UPPER(md.md5) AS md5,
        CASE md.extension
                WHEN ''
                THEN md.file_name
                ELSE md.file_name||'.'||md.extension
            END AS file_name,
        md.bytes AS file_size,
        po.package_id
    FROM
        METADATA AS md,
        PACKAGE_OBJECT AS po
    WHERE
        md.object_id = po.object_id
/* FILE(sha256,sha1,md5,file_name,file_size,package_id) */;
CREATE VIEW MFG AS
    SELECT
        manufacturer_id,
        name
    FROM
        MANUFACTURER
/* MFG(manufacturer_id,name) */;
CREATE VIEW OS AS
        SELECT
        os.operating_system_id,
        os.name,
        os.version,
        mos.manufacturer_id
    FROM
        OPERATING_SYSTEM AS os,
```

```
        MANUFACTURER_OPERATING_SYSTEM AS mos
    WHERE
        os.operating_system_id = mos.operating_system_id
/* OS(operating_system_id,name,version,manufacturer_id) */;
CREATE VIEW PKG AS
    SELECT
        a.package_id,
        a.name,
        COALESCE(a.version, a.build, a.latest_copyright, a.other) AS version,
        osa.operating_system_id,
        ma.manufacturer_id,
        l.name AS language,
        at.description AS application_type
    FROM
        APPLICATION AS a,
        OPERATING_SYSTEM_APPLICATION AS osa,
        MANUFACTURER_APPLICATION AS ma,
        APPLICATION_LANGUAGE AS al,
        LANGUAGE AS l,
        APPLICATION_APPLICATION_TYPE AS aat,
        APPLICATION_TYPE AS at
    WHERE
        a.application_id = osa.application_id
    AND
        a.application_id = ma.application_id
    AND
        a.application_id = al.application_id
    AND
        al.language_id = l.language_id
    AND
        a.application_id = aat.application_id
    AND
        aat.application_type_id = at.application_type_id
/* PKG(package_id,name,version,operating_system_id,manufacturer_id,language,application_type) */;
```

Minimal RDSv3 SQLite Database

There has been much interest by our users for including a minimal hash database version of the RDSv3 publication, which will reduce the size of the database and delta file downloads, by only including data that is equivalent to the old RDS 2.XX text files previously published by the NSRL. The minimal database publication mirrors the FILE, MFG, OS, and PKG views of the full publication database, but will only contain the set of FILE data that includes just distinct SHA256 hashes (reference the minimal database schema below).

Currently, the NSRL is only planning to publish the minimal database for the Modern hash set, as this set has received the most interest for the inclusion of a minimal database. The NSRL may publish minimal databases for other hash sets, if there is sufficient demand.

The minimal RDSv3 download will be available on the NSRL website's download page as a zip file. The contents of the zip file will depend on whether the publication is a full database release, or a delta release. In the full database publication, the zip file will contain the following:

- RDS_YYYY.MM.<patch>_<set>_minimal.db
    - (ex. RDS_2022.03.1_modern_minimal.db)
    - SQLite database publication of the minimal RDS set
- RDS_YYYY.MM.<patch>_<set>_minimal .schema.sql
    - (ex. RDS_2022.03.1.schema_minimal .sql)
    - Current schema for the minimal SQLite database publication
- readme.txt
    - Readme file containing information about the publication
- signatures.txt
    - SHA-1 hash signatures of the three above files published in the minimal RDS set

In the delta minimal RDSv3 publication, the zip file will contain the following:

- RDS_YYYY.MM.<patch>_<set>_minimal_delta.sql
    - Full set of SQL INSERT, UPDATE, and DELETE statements needed in order to update the previous minimal RDSv3 database file to the latest version of published data.
    - It is intended that the user open the previous minimal RDS database version in SQLite, and then run all statements in the given file. This can be done by using the following command while connected to the previous minimal release database:
        - .read RDS_YYYY.MM.<patch>_<set>_minimal_delta.sql
        - ex) .read RDS_2022.01.3_modern_minimal_delta.sql
    - This process will take a few minutes to complete.
    - Further update instructions can be found in this document below.
- RDS_YYYY.MM.<patch>_<set>.schema.sql (ex. RDS_2022.01.3.schema.sql)
    - Current schema for the SQLite database publication
- readme.txt

    - Readme file containing information about the delta publication
- signatures.txt
    - SHA-1 hash signatures of the above files to be published in the RDS set

For any and all questions, comments, or feedback regarding the transition to the RDSv3 publication, please contact us at nsrl@nist.gov .

For current information about the NSRL and RDS, please see https://www.nsrl.nist.gov or contact the project team at nsrl@nist.gov .

Updating the Minimal RDSv3 Database

Updating the minimal RDSv3 database publication can be done using the same steps outlined above in the document, for updating the normal RDSv3 database publication with a delta SQL file. Be sure to only update the minimal RDSv3 database with a minimal RDSv3 delta file of the same hash set.

Minimal RDSv3 SQLite Database Publication Schema

```
CREATE TABLE FILE (
 sha256     VARCHAR NOT NULL,
 sha1       VARCHAR NOT NULL,
 md5        VARCHAR NOT NULL,
 crc32      VARCHAR NOT NULL,
 file_name  VARCHAR NOT NULL,
 file_size  INTEGER NOT NULL,
 package_id INTEGER NOT NULL,
 CONSTRAINT PK_FILE__FILE PRIMARY KEY (sha256, sha1, md5, file_name, file_size, package_id)
);
CREATE TABLE MFG (
 manufacturer_id INTEGER NOT NULL,
 name            VARCHAR NOT NULL,
 CONSTRAINT PK_MFG__MFG_ID PRIMARY KEY (manufacturer_id)
);
CREATE TABLE OS (
 operating_system_id INTEGER NOT NULL,
 name                VARCHAR NOT NULL,
 version             VARCHAR NOT NULL,
 manufacturer_id     INTEGER NOT NULL,
 CONSTRAINT PK_OS__OS_ID PRIMARY KEY (operating_system_id, manufacturer_id)
);
CREATE TABLE PKG (
 package_id          INTEGER NOT NULL,
 name                VARCHAR NOT NULL,
 version             VARCHAR NOT NULL,
 operating_system_id INTEGER NOT NULL,
 manufacturer_id     INTEGER NOT NULL,
 language            VARCHAR NOT NULL,
 application_type    VARCHAR NOT NULL,
 CONSTRAINT PK_PGK__PKG_ID PRIMARY KEY (package_id, operating_system_id, manufacturer_id, language,
application_type)
);
CREATE TABLE VERSION (
 version      VARCHAR UNIQUE NOT NULL,
 build_set    VARCHAR NOT NULL,
 build_date   TIMESTAMP DEFAULT CURRENT_TIMESTAMP NOT NULL,
 release_date TIMESTAMP NOT NULL,
 description  VARCHAR NOT NULL,
 CONSTRAINT PK_VERSION__VERSION PRIMARY KEY (version)
);
CREATE VIEW DISTINCT_HASH AS
 SELECT DISTINCT
  sha256,
  sha1,
  md5,
  crc32
 FROM
  FILE
/* DISTINCT_HASH(sha256,sha1,md5,crc32) */;
```