Record and Event Stream File Formats

# [**Online Doc Link**](https://hederatest.sharepoint.com/:w:/s/Engineering/EZIuSuJ9eSZOkbCfu0HEa6AByPTaV0ixTfSXeDg7153IvQ?e=oonPwj)

# **Overview**

The Hedera network produces record stream files and event stream files that captures chronological information about transactions that took place on the network. A record stream file is a file that contains a series of transactions in chronological order that occurred within a two second interval. The record stream file (.rcd) contains a transaction record for each transaction in that file. For each record stream file there is a corresponding signature file (rcd\_sig) that includes the signature generated by the node. The event stream file(.evts) contains information about an event. Event information includes event hash data, event unhashed data, and consensus data. Each event stream files are created in 5 second intervals. Each event stream file has a corresponding signature file (.evts\_sig) that includes the node signature.

The record stream file and event stream file formats will both be updated to version 5 in a future mainnet release.

# **Version Migration**

|  |  |  |
| --- | --- | --- |
| File Type | Current Version | Next Version |
| Record Stream File | [2](#_Version_2_Record) | [5](#_Version_5_Record) |
| Record Stream Signature File | [No version number](#Version_2_Record_Sig) | [5](#Version_5_Record_Sig) |
| Event Stream File | [3](#_Version_3_Event) | [5](#_Version_5_Event) |
| Event Stream Signature File | [No version number](#Version_3_Event_Sig) | [5](#Version_5_Event_Sig) |

Note: Current rcd\_sig and evts\_sig file formats do not have a version number. The first byte's value is 4, which denotes a marker. For maintaining backwards compatibility, we made the first byte in the new version stream signature file be 5, which denotes the version. Thus, we use version 5 as the next version number.

## **Version 2 Record Stream File Format**

### **Record Files Names**

A record stream file name is made up of a string representation of the Instant of consensus timestamp of the first transaction in the file using ISO-8601 representation, with colons converted to underscores for Windows compatibility. The nano-of-second outputs zero, three, six or nine digits as necessary.

Examples of a record stream file name with corresponding signature file:

Example 1:

Record Stream File: *2020-10-19T21\_35\_39Z.rcd*

Record Stream Signature File*: 2020-10-19T21\_35\_39Z.rcd\_sig*

Example 2:

Record Stream File: *2020-10-19T21\_35\_39.454265Z.rcd*

Record Stream Signature File: *2020-10-19T21\_35\_39.454265Z.rcd\_sig*

### **Record File Format (.rcd) – Version 2**

The below table describes the content that can be parsed from a record file.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Record Stream File Format Version | int (4) | Record stream file format version,  Value: 2 |
| Hapi Version | int (4) | HAPI protocol version,  Value: 3 |
| Prev File Hash Marker | byte | Value: 1 |
| Prev File Hash | byte[48] | SHA384 hash of previous file, if not exist then all zeroes |
| Record Marker | byte | Value: 2 |
| Length of Transaction | int (4) | Byte size of the following Transaction message |
| Transaction | byte[] | Serialized Transaction message bytes |
| Length of TransactionRecord | int (4) | Byte size of the following TransactionRecord message |
| TransactionRecord | byte[] | Serialized TransactionRecord message bytes |
| Record Marker | byte | Value: 2 |
| ... | ... | ... |

The signature file \*.rcd\_sig signs the hash of corresponding \*.rcd file;

### **Record Signature File Format (.rcd\_sig) – Version 2**

### **Note: File format doesn’t have version number**

The below table describes the content that can be parsed from a record signature file.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| File Hash Marker | byte | Value: 4 |
| File Hash | byte[48] | SHA384 hash of corresponding \*.rcd file |
| Signature Marker | byte | Value: 3 |
| Length of Signature | int (4) | Byte size of the following signature bytes |
| Signature | byte[] | Signature bytes |

### **File Hash Calculation**

The record file hash is calculated using the following formula.

*h[i] = hash(p[i-1] || h[i-1] || hash(c[i-1]))*

*||* denotes concatenation

h[i] is the hash of the file i

p[i-1] is the contents in the file before the PREV\_FILE\_HASH

h[i-1] is the hash of file i-1, i.e., PREV\_FILE\_HASH

c[i-1] is the contents of the file after PREV\_FILE\_HASH

## **Version 5 Record Stream File Format**

Generated using functionality provided by swirlds-common in the swirlds-platform sdk for serializing objects to record files (.rcd), and deserializing objects from record file (.rcd). In the next release, swirlds-common package will contain functionalities for parsing and verifying version 5 record stream files, which can be used by both services and mirror node.

### **Record Files Names**

A record file name is created using a string representation of the Instant of consensus timestamp of the first transaction in the file using ISO-8601 representation, with colons converted to underscores for Windows compatibility.

The nano-of-second **always outputs nine** digits with padding when necessary, to ensure same length filenames and proper sorting.

Examples of a record stream file name with corresponding signature file:

Example 1:

Record File (.rcd): *2020-10-19T21\_35\_39.000000000Z.rcd*

Record Signature File(.rcd\_sig): *2020-10-19T21\_35\_39.000000000Z.rcd\_sig*

Example 2:

Record File (.rcd): *2020-10-19T21\_35\_39.454265000Z.rcd*

Record Signature File(.rcd\_sig): *2020-10-19T21\_35\_39.454265000Z.rcd\_sig*

### **Record File Format (.rcd) – Version 5**

The below tables describe the content that can be parsed from a record file.

*MD – Meta Data*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** | **MD** |
| Record Stream File Format Version | int (4) | Record stream file format version,  Value: 5 | y |
| Hapi Proto Major Version | int (4) | Matches NetworkGetVersionInfo.hapiProtoVersion  Major.Minor.Patch  Major version: 0;  Minor version: 9;  Patch version: 0 | y |
| Hapi Proto Minor Version | int (4) |
| Hapi Proto Patch Version | int (4) |
| Object stream version | int (4) | Value: 1  This defines the format of the remainder of the file. This version number is used when parsing a stream file with methods defined in swirlds-common package | y |
| Start Object Running Hash | byte[] | Running Hash of all RecordStreamObject before writing this file.  See `Hash object` table below for details | y |
| 1st RecordStreamObject | byte[] | Serialized RecordStreamObject bytes.  See ` RecordStreamObject ` table below for details | n |
| 2nd RecordStreamObject | byte[] | ... | n |
| ... | ... | ... | n |
| End Object Running Hash | byte[] | Running Hash of all RecordStreamObject before closing this file. | y |

Hash Object

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class ID | long (8) | Value: 0xf422da83a251741e |
| Class Version | int (4) | Value: 1  This version number will be updated each time when the serialization format of a Hash object is changed |
| Digest Type | int (4) | Value: 0x58ff811b  denotes SHA-384 |
| Length of Hash | int (4) | Value: 48 for SHA-384 |
| Hash Bytes | byte[] | Serialized Hash bytes |

RecordStreamObject

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class ID | long (8) | Value: 0xe370929ba5429d8b |
| Class Version | int (4) | Value: 1 |
| Length of TransactionRecord | int (4) | Byte size of the following TransactionRecord message |
| TransactionRecord | byte[] | Serialized TransactionRecord message bytes |
| Length of Transaction | int (4) | Byte size of the following Transaction message |
| Transaction | byte[] | Serialized Transaction message bytes |

### **Record Signature File Format (.rcd\_sig) – Version 5**

The below tables describe the content that can be parsed from a record signature file.

In version 5, the record stream signature file format is the same as event stream signature file format.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Signature File Format Version | int (4) | Value: 5 |
| Object Stream Signature Version | int (4) | Value: 1  This defines the format of the remainder of the signature file. This version number is used when parsing a signature file with methods defined in swirlds-common package |
| Entire Hash of the corresponding stream file | byte[] | Hash of the entire corresponding stream file |
| Signature on hash bytes of Entire Hash | byte[] | A signature object generated by signing the hash bytes of Entire Hash. See ` Signature ` table below for details |
| Metadata Hash of the corresponding stream file | byte[] | Metadata Hash of the corresponding stream file |
| Signature on hash bytes of Metadata Hash | byte[] | A signature object generated by signing the hash bytes of Metadata Hash |

Signature object

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class ID | long (8) | Value: 0x13dc4b399b245c69 |
| Class Version | int (4) | Value: 1 |
| SignatureType | int (4) | Value: 1  Denotes SHA384withRSA |
| Length of Signature | int (4) | Byte size of the signature bytes |
| Signature bytes | byte[] | Serialized Signature bytes | |

### **File Hash Calculation**

In version 5 there are 3 different hashes being calculated:

1. Object Running hash
   1. Object Running hash is calculated based on hash of each object, which allows us to remove contents of an object from a stream file while maintaining an unbroken chain of hashes.
   2. The object running hash will be saved in state, so that reconnect nodes will be able to continue generating stream files identical to those generated by other nodes after reconnecting.
   3. ***Object Running Hash = hash(Object Running Hash || hash(OBJECT))***

*||* denotes concatenation

In the event stream file OBJECT will be each event;

In the record stream file OBJECT will be the Record Stream Object;

1. Entire .rcd hash
   1. This hash is calculated on all the bytes of a .rcd file.
   2. With the entire hash, mirror node can download valid .rcd files whose entire hash are agreed on by valid signatures of at least 1/3 of nodes.
   3. Suppose the content of a .rcd file with version 5’s content is as following:

*f[i] = head[i] || startHash[i] || contents[i] || endHash[i]*

*||* denotes concatenation

*f[i] denotes the whole content of a .rcd file;*

*head[i]* denotes the bytes before the start Object Running Hash;

*startHash[i]* denotes the Object Running Hash before writing this file;

*contents[i]* denotes the content between the start Object Running Hash and the end Object Running Hash;

*endHash [i]* denotes the Object Running Hash before closing this file;

Entire hash is calculated as following:

**entireHash[i] = *hash(head[i] || startHash[i] || contents[i] || endHash[i])***

1. Metadata .rcd hash
   1. This hash is calculated on metadata bytes in the file, denoted in the table.
   2. If some contents of an object were removed from a file, the metadata hash would still be valid.
   3. Metadata hash is calculated as following:

***metaHash[i] = hash(head[i] || startHash[i] || endHash[i])***

## **Version 3 Event Stream File Format**

### **Event Stream Files Names**

A string representation of the Instant of consensus timestamp of the first transaction in the file using ISO-8601 representation, with colons converted to underscores for Windows compatibility. The nano-of-second outputs zero, three, six or nine digits as necessary.

Example of an event stream file name with corresponding signature file:

Event Stream File (.evts): *2020-10-19T21\_35\_39.454265Z.evts*

Event Stream Signature File (.evts\_sig): *2020-10-19T21\_35\_39.454265Z.evts\_sig*

### **Event Stream File Format (.evts) – Version 3**

The below table describes the content that can be parsed from an event stream file.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Event Stream File Format Version | int (4) | Value: 3 |
| Prev File Hash Marker | byte | Value: 1 |
| Prev File Hash | byte[48] | SHA384 hash of previous .evts file, if not exist then all zeroes |
| Event Marker | byte | Value: 90 |
| Event Format Version | int (4) | Value: 3 |
| Event | byte[] | serialized Event bytes  see ` ConsensusEvent` table for details |
| Event's Hash | byte[] | This Event’s Hash |
| Event Marker | byte | Value: 90 |
| ... | … | … |

### **Event Stream Signature File Format (.evts\_sig) – Version 3**

### **Note: File format doesn’t have a version number**

The below table describes the content that can be parsed from an event stream signature file.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| File Hash Marker | byte | Value: 4 |
| File Hash | byte[48] | SHA384 hash of corresponding \*.evts file |
| Signature Marker | byte | Value: 3 |
| Length of Signature | int (4) | Byte size of the following signature bytes |
| Signature | byte[] | Signature bytes |

## **Version 5 Event Stream File Format**

Generated using functionality provided by swirlds-common for serializing objects to .evts file, and deserializing objects from .evts file. In the next release, swirlds-common will contain functionalities for parsing and verifying version 5 event stream files, which can be used by both services and mirror node.

### **Files Names**

A string representation of the Instant of consensus timestamp of the first transaction in the file using ISO-8601 representation, with colons converted to underscores for Windows compatibility.

The nano-of-second **always outputs nine** digits with padding when necessary, to ensure same length filenames and proper sorting.

Example of an event stream file name with corresponding signature file.

Example 1:

Event Stream File (.evts): *2020-10-19T21\_35\_39.000000000Z.evts*

Event Stream Signature File (.evts\_sig): *2020-10-19T21\_35\_39.000000000Z.evts\_sig*

Example 2:

Event Stream File (.evts): *2020-10-19T21\_35\_39.454265000Z.evts*

Event Stream Signature File (.evts\_sig): *2020-10-19T21\_35\_39.454265000Z.evts\_sig*

### **Event Stream File Format (.evts) – Version 5**

The below tables describe the information that can be parsed from an event stream file.

*MD – Meta Data*

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** | **MD** |
| Event Stream File Format Version | int (4) | Event stream file format version,  Value: 5 | y |
| Object Stream Version | int (4) | Value: 1  This defines the format of the remainder of the file. This version number is used when parsing a stream file with methods defined in swirlds-common package | y |
| Start Object Running Hash | byte[] | Running Hash of all Events reached consensus before writing this file | y |
| 1st ConsensusEvent | byte[] | Serialized ConsensusEvent bytes.  The following ` ConsensusEvent ` table contains detail of a ConsensusEvent | n |
| 2nd ConsensusEvent | byte[] | ... | n |
| ... | ... | ... | n |
| End Object Running Hash | byte[] | Running Hash of all Events reached consensus before closing this file | y |

ConsensusEvent

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class ID | long (8) | Value: 0xe250a9fbdcc4b1ba |
| Class version | Int | Value: 1 |
| BaseEventHashedData | byte[] | Serialized BaseEventHashedData bytes |
| BaseEventUnhashedData | byte[] | Serialized BaseEventUnhashedData bytes |
| ConsensusData | byte[] | Serialized ConsensusData bytes |

BaseEventHashedData

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class Version | int (4) | Value: 1 |
| Creator’s Id | long (8) | ID of this event's creator |
| SelfParent's Generation | long (8) | the generation for the event’s self parent |
| OtherParent's Generation | long (8) | the generation for the event’s other parent |
| SelfParent's Hash  Class version | int (4) | Value: 1 |
| SelfParent's Hash  Digest Type | int (4) | Value: 0x58ff811b  denotes SHA-384 |
| SelfParent's Hash  length | int (4) | Value: 48 for SHA-384 |
| SelfParent's  Hash bytes | byte[] | SelfParent's Hash bytes |
| OtherParent's Hash  Class version | int (4) | Value: 1 |
| OtherParent's Hash  Digest Type | int (4) | Value: 0x58ff811b  denotes SHA-384 |
| OtherParent's Hash  length | int (4) | Value: 48 for SHA-384 |
| OtherParent's Hash  bytes | byte[] | OtherParent's Hash bytes |
| Time Created | byte[] | Creation time of this event  See `Instant` table for details |
| Transactions | byte[] | A serialized array of Transactions,  Details is shown in the following table |

Transactions: serialized form of an array of transactions

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Name** | **Type (Bytes)** | **Description** |
|  | Size of the Transaction Array | int (4) | Number of transactions in this array.  -1 denotes null.  0 denotes empty array.  Stop deserialization when size is –1 or 0. |
|  | All Same Class | boolean (1 bit) | Value: true |
| 1st  Transaction | isNull | boolean (1 bit) | Whether this transaction is null |
| Transaction Class Version | int (4) | Value: 1 |
| Length of Contents | int (4) | Length of contents in this transaction |
| Checksum | int (4) | 277 – contents.length |
| System | boolean (1 bit) | whether this transaction was originated by the application or the platform |
| Contents | byte[] | contents bytes |
| Size of Signatures List | int (4) | Value: 0  signatures are not serialized, so the size is 0 |
| CheckSum | int (4) | Value: 353 |
| 2nd  Transaction | ... | ... | ... |
| ... | ... | ... | ... |

Instant

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Epoch Second | long (8) | If this instant is null, Epoch Second will be Long.MIN\_VALUE |
| Nanos | long (8) | If the instant is null, Nanos will not be written |

BaseEventUnhashedData

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class Version | int (4) | Value: 1 |
| Creator Sequence | long (8) | sequence number for this by its creator (0 is first) |
| OtherParent’s ID | long (8) | ID of otherParent |
| OtherParent’s Sequence | long (8) | sequence number for otherParent event |
| Length of Signature | int (4) | length of signature bytes |
| Signature bytes | byte[] | creator's signature bytes for this event |

ConsensusData

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Class Version | int (4) | Value: 2 |
| Generation | long (8) | generation (which is 1 plus max of parents' generations) |
| Round Created | long (8) | the created round of this event |
| isStale | boolean (1 bit) | is there a consensus that this event is stale (no order, transactions ignored) |
| Last Event in Round Received | boolean (1 bit) | is this event the last in consensus order of all those with the same received round |
| Consensus Timestamp | byte[] | the community's consensus timestamp for this event  See `Instant` table for details |
| Round Received | long (8) | round where >=1/2 famous see this event |
| Consensus Order | long (8) | the event’s consensus order in history (0 first) |

### **Event Stream Signature File Format (.evts\_sig) – Version 5**

The below table describes the content that can be parsed from an event stream signature file.

In version 5, the event stream signature file format is the same as record stream signature file format.

|  |  |  |
| --- | --- | --- |
| **Name** | **Type (Bytes)** | **Description** |
| Signature File Format Version | int (4) | Value: 5 |
| Object Stream Signature Version | int (4) | Value: 1  This defines the format of the remainder of the signature file. This version number is used when parsing a signature file with methods defined in swirlds-common package |
| Entire Hash of the corresponding stream file | byte[] | Hash of the entire corresponding stream file |
| Signature on hash bytes of Entire Hash | byte[] | A signature object generated by signing the hash bytes of Entire Hash. See ` Signature ` table below for details |
| Metadata Hash of the corresponding stream file | byte[] | Metadata Hash of the corresponding stream file |
| Signature on hash bytes of Metadata Hash | byte[] | A signature object generated by signing the hash bytes of Metadata Hash |