ILDG Binary File Format (Rev. 1.2)

ILDG Metadata Working Group

November 10, 2024

This document specifies the file format which is used to exchange binary files within ILDG.

1 Overview

For sharing binary files which contain, e.g., gauge field configurations, ILDG adopted the strategy of defining a real file format. All providers of configurations have to provide their configurations using this file format. The goal was to to define a structured file format which can encapsulate, in an extensible way, binary data and meta data within a single file. While currently only exchange of binary files containing gauge field configurations is foreseen, in future also other kinds of binary data, e.g. different kinds of propagators, may be exchanged.

2 Specification

An ILDG binary file consists of several parts. The ILDG specification defines (i) the packaging of these parts into a single file and (ii) the format and content of each individual part. Currently, the packaging (i) is specified to only use the LIME file format, but may be extended to other formats in the future.

2.1 Packaging (LIME)

LIME stands for "Lattice QCD Interchange Message Encapsulation" and has been developed by SciDAC [1]. It allows encapsulation of one or more *records* containing ASCII or binary data. Records must be grouped into *messages.* A message can contain one or more records. Each record has a record type (string) and multiple records may have the same type. A LIME file is a concatenation of one or more messages. Thus, the concatenation of LIME files yields again a valid LIME file.

2.2 File structure (LIME)

An ILDG binary file is a sequence of LIME messages/records as illustrated in the following table:

message	record	record type
...	...	...
#n	...	...
	#i	ildg-format
	...	...
	#j	[ildg-update]
	...	...
	#k	ildg-binary-data
	...	...
...	...	...
#m	...	...
	#l	ildg-data-lfn
	...	...
...	...	...

The message/record structure of an ILDG binary file is subject to the following rules:

- Matching of the record type (string) is performed case sensitive.
- The first part of the record type string, i.e. all characters until the first hyphen ('-'), identifies a namespace.
- Messages/records indicated by '...' or in square brackets may be absent.
- Groups are free to introduce their own messages/records and insert them in the locations indicated by '...'.
- Each group introducing their own records should use a unique namespace. These namespaces must be different from the reserved namespace '`ildg-`' and should be registered with ILDG to avoid name clashes.
- The order of the messages `#n` and `#m` is not fixed and both messages could also be combined into a single message.
- The records `ildg-format` and `ildg-binary-data` must be contained within the same message and the order of the two records is fixed to `#i<#k`.
- If a file contains multiple messages with a record `ildg-binary-data` and the values of the `<field>` elements in any pair of `ildg-format` records are equal, then each message with a record `ildg-binary-data` must also contain a record `ildg-update`. The order of the three records is fixed to `#i<#j<#k`.

- The records `ildg-format`, `ildg-update`, and `ildg-data-lfn` may only contain printable ASCII characters (`0x20`. . . `0x7E`), horizontal tab (`0x09`), or newline (`0x0A`). A trailing null byte (`0x00`) is allowed, but not recommended. Any bytes following a null byte have no significance.

Every ILDG binary file is associated with the unique identifier that is specified in the record `ildg-data-lfn`. Within an ILDG binary file, each message containing an `ildg-binary-data` record must be uniquely identified by the combination of the value of the `<field>` element in the record `ildg-format` and the content of the record `ildg-update` (if present and required).

2.3 Record `ildg-format`

This record consists of an XML document which contains a **minimal** set of non-mutable parameters **needed to read the binary data**. The document has to conform to the schema provided in appendix A.1.

Example XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<ildgFormat xmlns="http://www.lqcd.org/ildg"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.lqcd.org/ildg/filefmt.xsd">
  <version> 1.2 </version>
  <field> su3gauge </field>
  <rows> 2 </rows>
  <precision> 32 </precision>
  <lx> 20 </lx> <ly> 20 </ly> <lz> 20 </lz> <lt> 64 </lt>
</ildgFormat>
```

Remarks:

- If in `ildg-binary-data` all $N_{row} = N_c$ rows of the links are stored, where N_c denotes the number of colours of the gauge field, then the element

  `<rows>` N_{row} `</rows>`

  is optional (for backward compatibility with Revision 1.1). Otherwise, if the links are stored with the last row(s) omitted, the element `<rows>` is required to specify the number of actually stored rows, N_{row}.

- The element `<precision>` specifies the number of bits which are used to store floating-point numbers in `ildg-binary-data`. This precision may be different (typically lower) than the precision specified in the config metadata (which refers to the internal storage precision used for the gauge field in the simulation algorithm).

- The values of the elements `<lx>`, …, `<lt>` specify the number of hyper-planes in which the sites have links associated in at least one of the positive directions.

  In case of `dirichlet` boundary conditions, e.g. in the t-direction, a lattice with values N_x, …, N_t in the elements `<lx>`, …, `<lt>`, is usually called a $N_x \times N_y \times N_z \times (N_t - 1)$ lattice in the literature. Thus, a value $N_t - 1$ (instead of N_t) might be used in the ensemble metadata.

2.4 Record `ildg-update`

This record contains a single string of digits. The string represents the update number of the configuration and must be identical to the value of the element `<update>` in the corresponding `markovStep` of the configuration metadata.

Presence of the record `ildg-update` (in all messages that contain an `ildg-binary-data` record) is optional if the fields in all `ildg-binary-data` records were generated by the *same* update[1], i.e. belong to the same Markov step.

2.5 Record `ildg-binary-data`

This record may contain different kinds of fields or other data. Currently, only storing of gauge fields is foreseen. The record consists of a sequence of IEEE floating point numbers. The precision is defined in the `ildg-format` record. The endianness is fixed to big.

2.5.1 Gauge configurations

A $U(N_c)$, $SU(N_c)$, $SO(N_c)$, or $Sp(N_c)$ gauge configuration is a set of $U(N_c)$, $SU(N_c)$, $SO(N_c)$, or $Sp(N_c)$ matrices, respectively, assigned to the links of a four-dimensional hypercube. If $U_\mu(n)$ denotes these link variables and $\chi(n)$ is a field transforming as a colour vector (e.g. a Dirac component of a quark field) at site $n = (n_1, n_2, n_3, n_4)$, then

$$\chi^\dagger(n) U_\mu(n) \chi(n+\hat{\mu}) \equiv \sum_{a,b=1}^{N_c} [\chi^\dagger(n)]_a [U_\mu(n)]_{ab} [\chi(n+\hat{\mu})]_b \tag{1}$$

is gauge invariant.

For $SU(N_c)$, and $U(N_c)$ with $N_c > 1$, and for $Sp(N_c)$ with $N_c \geq 4$ even, the gauge configuration $U_\mu(n)$ is stored as 8- (or 7-) dimensional array of floating point (complex) numbers. For $SO(N_c)$, the configuration is stored as 7-dimensional array of floating point numbers. For $U(1)$, the configuration is stored as 6- (or 5-) dimensional array of floating point (complex) numbers.

[1]For instance, if the file contains two `ildg-binary-data` records storing the $SU(3)$ and $U(1)$ gauge field from the same update of a QCD+QED simulation, the `ildg-update` record in the corresponding messages is optional. On the other hand, if the file contains multiple messages with gauge fields from different updates (e.g. $SU(3)$ fields from a QCD simulation), then the `ildg-update` record is mandatory in each such message.

The ordering of the array dimensions from slowest to fastest running index (with C-style index range in parentheses) is:

1. Site index in time-direction t ($0, \ldots, N_t - 1$; omitted for $N_t = 1$).
2. Site index in space-direction z ($0, \ldots, N_z - 1$; omitted for $N_z = 1$).
3. Site index in space-direction y ($0, \ldots, N_y - 1$; omitted for $N_y = 1$).
4. Site index in space-direction x ($0, \ldots, N_x - 1$; omitted for $N_x = 1$).
5. Link direction index μ ($0, \ldots, N_{dir} - 1$, where `0`$\leftrightarrow x$, `1`$\leftrightarrow y$, `2`$\leftrightarrow z$, `3`$\leftrightarrow t$).
6. Colour index a ($0, \ldots, N_{row} - 1$; omitted for $N_{row} = 1$).
7. Colour index b ($0, \ldots, N_c - 1$; omitted for $N_c = 1$).
8. Index referencing the real (0) or imaginary (1) part (omitted for $SO(N_c)$ and for $U(1)$ field stored as `u1phase`).

The corresponding declaration of such an array in C (i.e. the rightmost index runs fastest) is:

`double U[`N_t`] [`N_z`] [`N_y`] [`N_x`] [`N_{dir}`] [`N_{row}`] [`N_c`] [2];`

For FORTRAN (i.e. the leftmost index runs fastest) the equivalent statement is:

`complex U(`N_c`, `N_{row}`, `N_{dir}`, `N_x`, `N_y`, `N_z`, `N_t`)`

The array dimensions N_t, N_z, N_y, N_x, N_{row}, and N_c are determined by the values of the corresponding elements `<lt>`, `<lz>`, `<ly>`, `<lx>`, `<rows>`, and `<field>`, respectively, of the `ildg-format` record.

The value of N_{dir} is given by the number of non-trivial lattice directions, i.e. directions with more than one site. For $N_{dir} < 4$, ascending values of the direction index $\mu = 0, \ldots, N_{dir} - 1$ correspond to the directions in the sequence (x, y, z, t) after removing the trivial directions.

Optionally reduced storage

By default $N_{row} = N_c$. In reduced storage format[2], however, N_{row} may be equal to $N_c - 1$ for $SO(N_c)$ and $SU(N_c)$, or equal to $N_c/2$ for $Sp(N_c)$. The value of N_{row} must then be explicitly specified by the element `<rows>` in `ildg-format`.

[2]For $SU(3)$ and $N_{row} = 2$, the data layout of `ildg-binary-data` is identical to the "Lattice QCD Archive Format" [2].

Unphysical links for `open` or `dirichlet` boundary conditions

For `open` or `dirichlet` boundary conditions in one or more directions, **no** links in the corresponding direction(s) exist for sites on the last slice in the corresponding direction(s). These "unphysical" links are nevertheless stored in the (regular) layout of `ildg-binary-data`, and all variables of these links must be set to the following specific value[3]:

- `-NaN` (all bits 1) in the special case of an `ildg-binary-data` record with format `u1phase` (i.e. only if this value is specified in the `<field>` element in the corresponding `ildg-format` record)
- `+0.0` (all bits 0) in *all* other cases.

2.6 Record `ildg-data-lfn`

This record[4] contains a single string with the logical filename (LFN) used for storing the binary file in the Grid. The content must be identical to the element `<dataLFN>` of the corresponding configuration metadata document.

[3]Such byte patterns can not occur for any physical link. Moreover, they are simple to detect in the `ildg-binary-data` record and invariant under byte swapping.

[4]The `ildg-data-lfn` record is not necessarily part of the same LIME message which contains `ildg-binary-data`. Thus, the latter can be packed even when the LFN is not yet known, and an ILDG-compliant file can be obtained later on by concatenating a message containing the `ildg-data-lfn` record.

Appendix

A.1 Schema for XML content of the `ildg-format` record

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.lqcd.org/ildg"
           xmlns:xs="http://www.w3.org/2001/XMLSchema"
           xmlns="http://www.lqcd.org/ildg"
           elementFormDefault="qualified"
           attributeFormDefault="unqualified">

  <xs:simpleType name="fieldType">
    <xs:restriction base="xs:token">
      <xs:pattern value="s[ou][2-9]gauge"/>
      <xs:pattern value="s[ou][1-9][0-9]+gauge"/>
      <xs:pattern value="sp[468]gauge"/>
      <xs:pattern value="sp[1-9][0-9]*[02468]gauge"/>
      <xs:pattern value="u1phase"/>
      <xs:pattern value="u[1-9][0-9]*gauge"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:simpleType name="precisionType">
    <xs:restriction base="xs:NMTOKEN">
      <xs:enumeration value="32"/>
      <xs:enumeration value="64"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:element name="ildgFormat">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="version" type="xs:string"/>
        <xs:element name="field" type="fieldType"/>
        <xs:element name="rows" type="xs:integer" minOccurs="0"/>
        <xs:element name="precision" type="precisionType"/>
        <xs:element name="lx" type="xs:integer"/>
        <xs:element name="ly" type="xs:integer"/>
        <xs:element name="lz" type="xs:integer"/>
        <xs:element name="lt" type="xs:integer"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

A.2 Reconstruction of links stored in reduced formats

- Any matrix in $Sp(2N) \equiv Sp(2N,\mathbb{C}) \cap SU(2N)$ has the block form

$$\begin{pmatrix} A & B \\ -B^* & A^* \end{pmatrix} \tag{2}$$

  where A^* and B^* denotes the complex conjugate of the corresponding $N \times N$ blocks (which are subject to further constraints). The last N rows of the full matrix in eq. (2) can then be reconstructed from the first N rows by toggling the corresponding signs (without any other arithmetic operations).

- The last row of an $SO(N)$ or $SU(N)$ matrix is the contraction of the first rows with the Levi-Civita symbol. Explicitly, if $u^{(k)}$ denotes the k-th row vector, the elements of the N-th row can be reconstructed as

$$u_k^{(N)} = \sum_{i_1,\ldots,i_{N-1}} \left[u_{i_1}^{(1)}\, u_{i_2}^{(2)} \cdots u_{i_{N-1}}^{(N-1)}\right]^* \cdot \epsilon_{i_1\, i_2 \ldots i_{N-1}\, k} \tag{3}$$

  with

$$\epsilon_{i_1\, i_2 \ldots i_{N-1}\, k} = \begin{cases} +1 & \text{if } (i_1,\, i_2, \ldots, i_{N-1}, k) \text{ is an even permutation of } (1,2,\ldots,N) \\ -1 & \text{if } (i_1,\, i_2, \ldots, i_{N-1}, k) \text{ is an odd permutation of } (1,2,\ldots,N) \\ 0 & \text{otherwise} \end{cases}$$

- For $U(N)$ and `u1phase` in the `<field>` element of the `ildg-format` record, only the (real) phase φ of the complex link variables $\exp(i\varphi)$ is stored.

In finite precision arithmetics the above specification of the reconstruction for $SO(N)$ and $SU(N)$ is not unique at the bit level, because addition operations are not associative and because $+0$ and -0 have different representations (e.g. `0x00000000` and `0x80000000`). If a unique reconstruction even at the bit level is required, the following additional conventions shall be employed:

- computations follow the IEEE Standard for Floating-Point Arithmetic (IEEE 754)
- results which are numerically zero are forced to be represented as $+0$ (all bits 0)
- eq. (3) is computed by first forming the products (which in the complex case and for $N \geq 4$ also involve repeated additions) from left to right,

$$\left(\cdots\left(([u_{i_1}^{(1)}\, u_{i_2}^{(2)})u_{i_3}^{(3)}\right)\cdots u_{i_{N-1}}^{(N-1)}\right)^*$$

  and then summing these terms according to lexicographic order of the index strings $(i_1,\, i_2,\, \ldots,\, i_{N-1})$, e.g. for $N=4$

$$u_4^{(4)} = \left[\ldots\left[(u_1^{(1)}\, u_2^{(2)})u_3^{(3)})^* - (u_1^{(1)}\, u_3^{(2)})u_2^{(3)})^*\right] + \cdots\right] - (u_3^{(1)}\, u_2^{(2)})u_1^{(3)})^*$$

References

[1] SciDAC Software Coordinating Committee, “LIME (Version 1.1),” `http://www.physics.utah.edu/~detar/scidac`

[2] ”Lattice QCD Archive Format”, V1.02, October 25, 1998, retrieved from `https://web.archive.org/web/20010608114140/http://qcd.nersc.gov/utilities/format-v103/node4.html`