university of groningen

# ILDG Metadata

## ILDG Hands-on Workshop


**Dirk Pleiter**

Bernoulli Institute

09 July 2025

# Overview

1. Overview and Requirements
2. Digression: XML Technologies
3. Ensemble Metadata
4. Configuration Metadata
5. Binary File Format
6. Concluding Remarks

# Content

# Metadata Content

1. **Physics**
   Description of action and simulation parameters
2. **Algorithm**
   Information about the used algorithm and algorithmic parameters
3. **Source code**
   Used code, compile time parameters, compilation software
4. **Machine**
   Machine used to produce the configuration
5. **Data management**
   Information related to data provenance and checksums

# Metadata Requirements

- **Standardised** description of the data (=**metadata**)
  - Solution strategy: **XML documents** which conform to an **XML schema**
- **Unique** description of the data
  - Avoid situation that action is described in two different ways,

    e.g. Iwasaki gauge action

    $$S = \beta \ (c_0 \,\text{plaquette} + c_1 \,\text{rectangular})$$

    or

    $$S = \tilde{c}_0 \,\text{plaquette} + \tilde{c}_1 \,\text{rectangular}$$
- Forward compatible **extensibility** of the standard
  - A metadata document conforming to the current version of the standard should also conform to its future versions
- Broad **coverage** of data that can be described
  - In parts achieved by foreseeing flexibility

# Example (1/2)

- SU(3) Wilson plaquette action with gluon fields $U$ and coupling $\beta$:

$$S_{\mathrm{G}} = \beta \sum_{\text{plaquette}} \frac{1}{3} \mathrm{Re}\ \mathrm{Tr}\,(1 - U_{\text{plaquette}})$$

# Example (2/2)

- (Almost) ILDG-compliant mark-up:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <plaquetteGluonAction>
  ...
    <gluonField>
      <gaugeGroup>SU(3)</gaugeGroup>
      ...
    </gluonField>
    <beta>5.2</beta>
  </plaquetteGluonAction>
  ...
</markovChain>
```

# Ensembles and Configurations

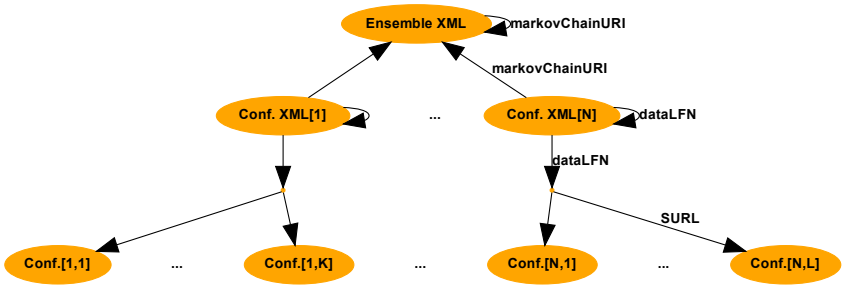
- **Aim:** Avoid replication of complicated data structures
- Markov chains are used to generate gauge field configurations

  Markov chain $\leftrightarrow$ Ensemble XML

  Markov step $\leftrightarrow$ Configuration XML

# Linking Metadata and Data

| Objects | Links |
|---|---|
| Ensemble XML document | markovChainURI |
| Configuration XML document | dataLFN |
| Binary data file | |



# Content

# Why XML?

- Both human-readable and machine-readable
- Standardized by the World Wide Web Consortium's (W3C) XML 1.0 Specification
- Availability of the XML schema (XSD) technology that allows to define the necessary metadata for interpreting and validating XML documents
- Availability of standardised technologies to query XML documents
  - W3C's XPath (current version: XPath 3.1)
  - W3C's XQuery (current version: XQuery 3.1)
  - *Note: The ILDG MDC currently only supports XPath 1.0*
- Many tools for processing XML documents are available
  - For parsing: `libxml2` with interfaces for C, C++, Python, Perl
  - For formatting and validation: `xmllint`

# Background on XML

- Simple example:

```
<?xml version="1.0" encoding="UTF-8"?>
<dataEntry>
  <key>7</key>
  <value>Hello world!</value>
</dataEntry>
```

- Key elements:
  - Tag: Markup construct that begins with < and ends with >
    - Start tag: <elem>
    - End tag: </elem>
    - Empty-element tag: <elem/>
  - Element: Logical document component starting/ending with a tag (or empty-element tag)
  - Attribute

# XML Schema

- Technology to formally describe the elements in an XML document
- Simple example:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="dataEntry">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="key" type="xs:decimal"/>
        <xs:element name="value" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# Validating XML Files (1/2)

- Document validation using the `xmllint` tool:

```
% xmllint --schema ./tst-schema.xsd ./tst-doc-ok.xml
<?xml version="1.0" encoding="UTF-8"?>
<dataEntry>
  <key>7</key>
  <value>Hello world!</value>
</dataEntry>
./tst-doc-ok.xml validates
```

# Validating XML Files (2/2)

- Try document with incorrect tag:

```
% xmllint --schema ./tst-schema.xsd ./tst-doc-bad-1.xml
<?xml version="1.0" encoding="UTF-8"?>
<dataEntry>
  <idx>7</idx>
  <value>Hello world!</value>
</dataEntry>
./tst-doc-bad-1.xml:3: element idx: Schemas validity error : Element 'idx': This element is not expected. Expected is ( key
).
./tst-doc-bad-1.xml fails to validate
```

- Try document with incorrect element content:

```
% xmllint --schema ./tst-schema.xsd ./tst-doc-bad-2.xml
<?xml version="1.0" encoding="UTF-8"?>
<dataEntry>
  <key>seven</key>
  <value>Hello world!</value>
</dataEntry>
./tst-doc-bad-2.xml:3: element key: Schemas validity error : Element 'key': 'seven' is not a valid value of the atomic type 'xs:decimal'.
./tst-doc-bad-2.xml fails to validate
```

# Tools for Creating XML Schemata: LLM

- Prompt submitted to DeepSeek (on 2025-07-08)

```
Provide an XML schema for the following XML document:
<?xml version="1.0" encoding="UTF -8"?>
<dataEntry>
  <key>7</key>
  <value>Hello world!</value>
</dataEntry>
```

- Response:

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="dataEntry">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="key" type="xs:integer"/>
        <xs:element name="value" type="xs:string"/>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

# Content

# Ensemble XML Overview

Top-level elements of the ensemble XML file:

- `markovChainURI`
  - A globally unique identifier that identifies
    - An Ensemble XML document
    - A set of configurations (metadata + data)
- `management`
  - Data relevant for tracing and recording the origins of the data (data provenance)
- `license`
- `funding`
- `physics`
  - Area where providers must provide all information needed to define the physical parameters of the data set (ensemble)
- `algorithm`
  - Area where providers can provide all information needed to describe the used algorithms (except for configuration-specific information)

# Ensemble XML: markovChainURI

- Schema mandates an Uniform Resource Identifier (URI)
- ILDG convention not enforced by schema to ensure uniqueness: mc://<reg>/<col>/<prj>/...
  - Regional grid (reg)
  - Collaboration (col)
  - Project (prj)
- Warning: ILDG currently does not have guaranteed mechanisms to check uniqueness of the markovChainURI

Example:

```
<?xml version="1.0"?>
<markovChain>
  <markovChainURI>
    mc://ldg/qcdsf/clover_nf2p1/b7p20kp12450kp12450-16x32
  </markovChainURI>
  ...
</markovChain>
```

# Ensemble XML: `management`

- Collaboration
  - String unconstrained by the schema, but should be consistent and unique
- Project name
  - String unconstrained by the schema, but should be consistent and unique at collaboration level
- Archive history
  - List of entries for tracking data provenance
  - Each entry comprises
    - Action: `add`, `replace`, `remove`
    - Participant identified either by
      - ORCID
        (plus optionally: name and/or institution)
      - name and institution
    - Date

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <management>
    <collaboration>qcdsf</collaboration>
    <projectName>clover_nf2p1</projectName>
    <archiveHistory>
      <archiveEvent>
        <revisionAction>add</revisionAction>
        <participant>
          <name>Yoshifumi Nakamura</name>
          <institution>NIC/DESY</institution>
        </participant>
        <date>2007-03-30T14:34:29+02:00</date>
      </archiveEvent>
    </archiveHistory>
  </management>
  ...
</markovChain>
```

# Ensemble XML: `license`

- Reminder FAIR principle R1.1:
  *"(Meta)data are released with a clear and accessible data usage license"*
  ☞ Defining a license has become mandatory
- Choice:
  - `licenseName`, `licenseURI`: Refer to an existing license by name and/or URL
    - Recommended: Creative Commons
  - Provide a custom license
- `embargoEndDate`: Date after which license applies (optional)
- `acknowledgments`: Optional list of elements `<citation>`

Example existing license:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <license>
    <licenseURI>
      https://creativecommons.org/licenses/by/4.0/
    </licenseURI>
    <embargoEndDate>2024-10-19</embargoEndDate>
  </license>
  ...
</markovChain>
```

Example custom license:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <license>
    <customLicenseText>
      All rights reserved
    </customLicenseText>
    <acknowledgments>
      <citation>
        doi:10.12345/arXiv.0000.0000
      </citation>
    </acknowledgments>
  </license>
  ...
</markovChain>
```

# Digression: Creative Commons (CC) Licenses

- Using the CC licenses is a standardised way to grant the public permission to use creative work under copyright law
- Various license options
  - Create Commons offers a tool for selecting a license
- Good choice: CC BY 4.0
  - **Share**: User is free to copy and redistribute the material in any medium or format for any purpose, even commercially
  - **Adapt**: User can remix, transform, and build upon the material for any purpose, even commercially
  - **Attribution**: Data user must give appropriate credit, provide a link to the license, and indicate if changes were made

# Ensemble XML: `funding`

- Option to add a non-empty list of funding references
- `fundingReference`:
  - `funderName`: Name of the funding agency
  - `awardTitle`: Title of a grant (optional)
  - `awardNumber`: Grant number (optional)
- Note: The markup is compliant to the DataCite schema

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <fundingReferences>
    <fundingReference>
      <funderName>
        European Commission
      </funderName>
      <awardTitle>
        XYZ Project
      </awardTitle>
      <awardNumber>
        101057437
      </awardNumber>
    </fundingReference>
  </fundingReferences>
  ...
</markovChain>
```

# Ensemble XML: `physics` / Overview

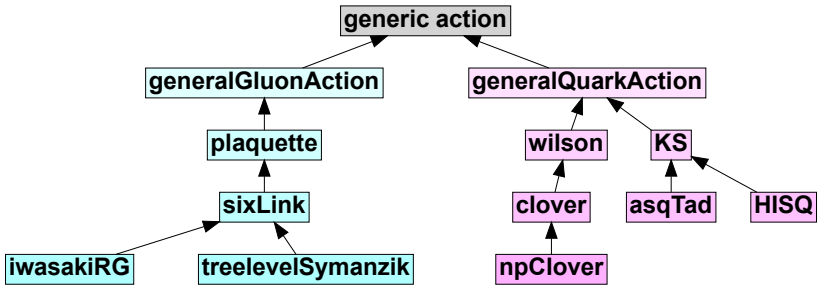
- `size`: Lattice size
  - List of dimensions: $(x, y, z, t)$ or $(t, x, y, z)$
  - $y$ and $z$ are optional
- `action/gluon`: Gluonic part of the action
- `action/quark`: Fermionic part of the action

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    <size>
      <x>16</x>
      <y>16</y>
      <z>16</z>
      <t>32</t>
    </size>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: `physics` / Action

- The description of action parameters is most critical to ensure metadata being unique and extensible
- Strategy: Hierarchy of actions



# Ensemble XML: `physics` / Gluonic Part (1/2)

- The element `gluon` contains an unbounded list of gluon actions
- Various gluon actions have been defined since the schema has been established
  - `plaquetteGluonAction`
  - `iwasakiRGGluonAction`
  - `tpLuescherWeiszGluonAction`
  - `anisotropicTpWilsonGluonActionType`
  - `...`

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    ...
    <action>
      <gluon>
        <plaquetteGluonAction>
          <glossary>http://...</glossary>
          <gluonField>
            <gaugeGroup>SU(3)</gaugeGroup>
            <representation>fundamental</representation>
            <boundaryCondition>
              <x>periodic</x>
              <y>periodic</y>
              <z>periodic</z>
              <t>periodic</t>
            </boundaryCondition>
          </gluonField>
          <beta>7.2</beta>
        </plaquetteGluonAction>
      </gluon>
      ...
    </action>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: `physics` / Gluonic Part (2/2)

- All gluon actions must contain
  - `glossary`: URI to documentation
  - `gluonField`
  - `topologyFixing` (optional)
- The element `gluonField` must contain the following sub-elements
  - `gaugeGroup`
    - Allowed values: SU(3), SU(2), ...
  - `representation`:
    - Allowed values: fundamental, adjoint
  - `boundaryCondition`
    - List of elements with the following values: periodic, antiperiodic, dirichlet, open, ...
- Other elements (e.g., `beta`) are action specific

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    ...
    <action>
      <gluon>
        <plaquetteGluonAction>
          <glossary>http://...</glossary>
          <gluonField>
            <gaugeGroup>SU(3)</gaugeGroup>
            <representation>fundamental</representation>
            <boundaryCondition>
              <x>periodic</x>
              <y>periodic</y>
              <z>periodic</z>
              <t>periodic</t>
            </boundaryCondition>
          </gluonField>
          <beta>7.2</beta>
        </plaquetteGluonAction>
      </gluon>
      ...
    </action>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: physics / Fermionic Part (1/3)

- The element quark contains an unbounded list of quark actions
- Various quark actions have been defined since the schema has been established
  - npCloverQuarkAction
  - overlapQuarkAction
  - ...
- All quark actions must contain
  - glossary: URI to documentation
  - quarkField
  - numberOfFlavours
  - linkSmearing (optional)
- Other elements are action specific

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    ...
    <action>
      ...
      <quark>
        <npCloverQuarkAction>
          <glossary>...</glossary>
          <quarkField>...</quarkField>
          <numberOfFlavours>2</numberOfFlavours>
          <linkSmearing>...</linkSmearing>
          <kappa>0.12450</kappa>
          <cSW>1.0</cSW>
        </npCloverQuarkAction>
        <npCloverQuarkAction>
          <glossary>...</glossary>
          <quarkField>...</quarkField>
          <numberOfFlavours>1</numberOfFlavours>
          <linkSmearing>...</linkSmearing>
          <kappa>0.12450</kappa>
          <cSW>1.0</cSW>
        </npCloverQuarkAction>
        ...
      </action>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: `physics` / Fermionic Part (2/3)

- The element `quarkField` must contain the following sub-elements
  - `normalisation`
    - Allowed values are not prescribed
  - `boundaryCondition`
    - List of elements with the following values: periodic, antiperiodic, dirichlet

Example for `quarkField`:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    ...
    <action>
      ...
      <quark>
        <npCloverQuarkAction>
          ...
          <quarkField>
            <normalisation>sqrt2kappa</normalisation>
            <boundaryCondition>
              <elem>periodic</elem>
              <elem>periodic</elem>
              <elem>periodic</elem>
              <elem>antiperiodic</elem>
            </boundaryCondition>
          </quarkField>
          ...
        </npCloverQuarkAction>
        <npCloverQuarkAction>
          ...
        </npCloverQuarkAction>
        ...
      </action>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: `physics` / Fermionic Part (3/3)

- The element `linkSmearing` must contain the following sub-elements
  - An element defining the link blocking, e.g.
    - `apeLinkBlocking`
    - `anisotropicApeLinkBlocking`
  - An element defining the link unitarization, e.g.
    - `stoutLinkUnitarization`
    - `invSqRootLinkUnitarization`
  - `numSmear`: integer value

Example for `linkSmearing`:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <physics>
    ...
    <action>
      ...
      <quark>
        <npCloverQuarkAction>
          ...
          <linkSmearing>
            <apeLinkBlocking>
              <rho0>0.1</rho0>
              <rho1>0.1</rho1>
            </apeLinkBlocking>
            <linkUnitarization>stout</linkUnitarization>
            <numSmear>1</numSmear>
          </linkSmearing>
          ...
        </npCloverQuarkAction>
        ...
     </action>
    ...
  </physics>
  ...
</markovChain>
```

# Ensemble XML: `algorithm`

- The element `algorithm` must contain the following sub-elements
  - `name`: A string restricted to legal XML 1.0 names
  - `glossary`: URI to documentation
  - `reference`: String containing, e.g., a publication
  - `exact`: Boolean value
  - `reweightingNeeded`: True if re-weighting is required
  - `parameters` (optional): list of elements `<parameter>`
  - Option to insert other XML elements using a different namespace
- `parameter` is a list of pairs
  - `name`: A string restricted to legal XML 1.0 names
  - `value`: Any simple type

Example:

```
<?xml version="1.0"?>
<markovChain>
  ...
  <algorithm>
    <name>qcdsfAcceleratedHMC</name>
    <glossary>
      doi:10.1016/j.nuclphysb.2013.05.019
    </glossary>
    <reference/>
    <exact>true</exact>
    <parameters>
      <parameter>
        <name>timeScaleRatio</name>
        <value>3</value>
      </parameter>
    </parameters>
  </algorithm>
</markovChain>
```

# Content

# Configuration XML Overview

Top-level elements of the configuration XML file:

- `dataLFN`
- `management`
- `implementation`
- `algorithm`
- `precision`
- `markovSequence`

Example:

```xml
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    ...
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    ...
  </markovSequence>
</gaugeConfiguration>
```

# Configuration XML: `management`

- Archive history
  - List of entries for tracking data provenance
  - Each entry comprises
    - Action: `generate`, `add`, `replace`, `remove`
    - Participant identified either by
      - ORCID
        (plus optionally: name and/or institution)
      - name and institution
    - Date

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    <archiveHistory>
      <archiveEvent>
        <revisionAction>generate</revisionAction>
        <participant>
          <name>Hinnerk Stueben</name>
          <institution>ZIB</institution>
        </participant>
        <date>2004-11-15T19:33:55+01:00</date>
      </archiveEvent>
    </archiveHistory>
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    ...
  </markovSequence>
</gaugeConfiguration>
```

# Configuration XML: `implementation`

- `machine`
  - `name`: String referring to the name of the computer used to generate the configuration
  - `institution`: String with the name of the organisation hosting that computer
  - `machineType`: String showing the computer type
  - `annotation`: String with optional additional information
- `code`
  - `name`: String with the name of the code
  - `version`: String with the code version
  - `date`: Date of the code release
  - `annotation`: String with optional additional information

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    ...
  </management>
  <implementation>
    <machine>
      <name>sr8000</name>
      <institution>LRZ Munich</institution>
      <machineType>Hitachi SR8000-F1</machineType>
    </machine>
    <code>
      <name>BQCD</name>
      <version>3.0.1</version>
      <date>2003-11-28T13:00:00+01:00</date>
    </code>
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    ...
  </markovSequence>
</gaugeConfiguration>
```

# Configuration XML: `algorithm`

- The optional element `parameters` comprises a list of elements <parameter>
- `parameter` is a pair:
  - `name`: A string restricted to legal XML 1.0 names
  - `value`: Any simple type
- It is allowed to insert other XML elements using a different namespace

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    ...
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    <parameters>
      <parameter>
        <name>targetResidue</name>
        <value>1e-07</value>
      </parameter>
    </parameters>
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    ...
  </markovSequence>
</gaugeConfiguration>
```

# Configuration XML: `precision`

- Allowed values:
  - `single`: Single precision calculations
  - `double`: Double precision calculations
  - `mixed`: Mix of single and double precision calculations
- Note that configurations may be stored in different precision

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <management>
    ...
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>double</precision>
  <markovStep>
    ...
  </markovStep>
</gaugeConfiguration>
```

# Configuration XML: `markovSequence`

- `markovChainURI`: Reference to the ensemble
- `series`: Whitespace-replaced and collapsed string identifying a run
- `markovStep`: An unbounded list of Markov steps
  - This new feature allows packing multiple configurations

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    ...
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    <markovChainURI>
      mc://...
    </markovChainURI>
    <series>563</series>
    <markovStep>
      ...
    </markovStep>
    <markovStep>
      ...
    </markovStep>
  </markovSequence>
</gaugeConfiguration>
```

# Configuration XML: `markovStep`

- `update`: Any simple type that references an update within a run
- `record/field`: Type of fields
- `record/crcCheckSum`: CRC32 checksum of the `ildg-binary-data` part of an ILDG configuration based on the LIME format
- `record/avePlaquette`: Average plaquette value (double)

Example:

```
<?xml version="1.0"?>
<gaugeConfiguration>
  <dataLFN>...</dataLFN>
  <management>
    ...
  </management>
  <implementation>
    ...
  </implementation>
  <algorithm>
    ...
  </algorithm>
  <precision>...</precision>
  <markovSequence>
    <markovChainURI>mc://...</markovChainURI>
    <series>563</series>
    <markovStep>
      <update>1310</update>
      <record>
        <field>su3gauge</field>
        <crcCheckSum>3806090226</crcCheckSum>
        <avePlaquette>0.5610635491</avePlaquette>
      </record>
    </markovStep>
  </markovSequence>
</gaugeConfiguration>
```

# References

- Schemata available at `https://gitlab.desy.de/ildg/mdwg/qcdml`
- Note the large collection of test XML documents
  - `ensemble/2.0.0/ref-valid`
  - `config/2.0.0/ref-valid`

# Content

# ILDG Binary File Format

- An ILDG binary file consists of several parts
- Scope of the ILDG binary file format specification
  - Packaging format: “Lattice QCD Interchange Message Encapsulation” (LIME)
  - Format and content of each individual parts
- Specification available here:
  `https://gitlab.desy.de/ildg/mdwg/file-format.git`

# File Format: Structure

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

# File Format: Record `ildg-format`

- XML document which contains a minimal set of non-mutable parameters needed to read the binary data
- Example:

```
<?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>
```

- New in ILDG 2.0: Now also field types other than `su3gauge` are also supported

# File Format: Record `ildg-binary-data`

- Currently only storing of gauge fields is foreseen
- Fields are stored as IEEE floating-point numbers
  - Precision defined in the `ildg-format` record
  - Endianness is fixed to big
- Ordering of the field array dimensions:

$$\texttt{double U[}N_t\texttt{][}N_z\texttt{][}N_y\texttt{][}N_x\texttt{][}N_{\mathrm{dir}}\texttt{][}N_{\mathrm{row}}\texttt{][}N_{\mathrm{c}}\texttt{][2];}$$

  - New in ILDG 2.0: $N_{\mathrm{row}} = N_{\mathrm{c}}$ or $N_{\mathrm{row}} = N_{\mathrm{c}} - 1$

# Content

# Concluding Remarks

- The ILDG metadata standards have been a robust framework for describing LQCD configurations data
  - Forward compatibility has been important
  - But: Evolution of the standard stagnated during the last 10 years
  - With the changes in ILDG 2.0 the schema is up-to-date again!
- The complexity of the underlying metadata schema is in practice not exposed to end-users
  - Most metadata documents are derived from existing documents with small modifications
- The binary file format standard has been updated in a backward compatible manner
  - Now the packing of many configurations within a single file is possible