OAK RIDGE National Laboratory

GNDS support in AMPX

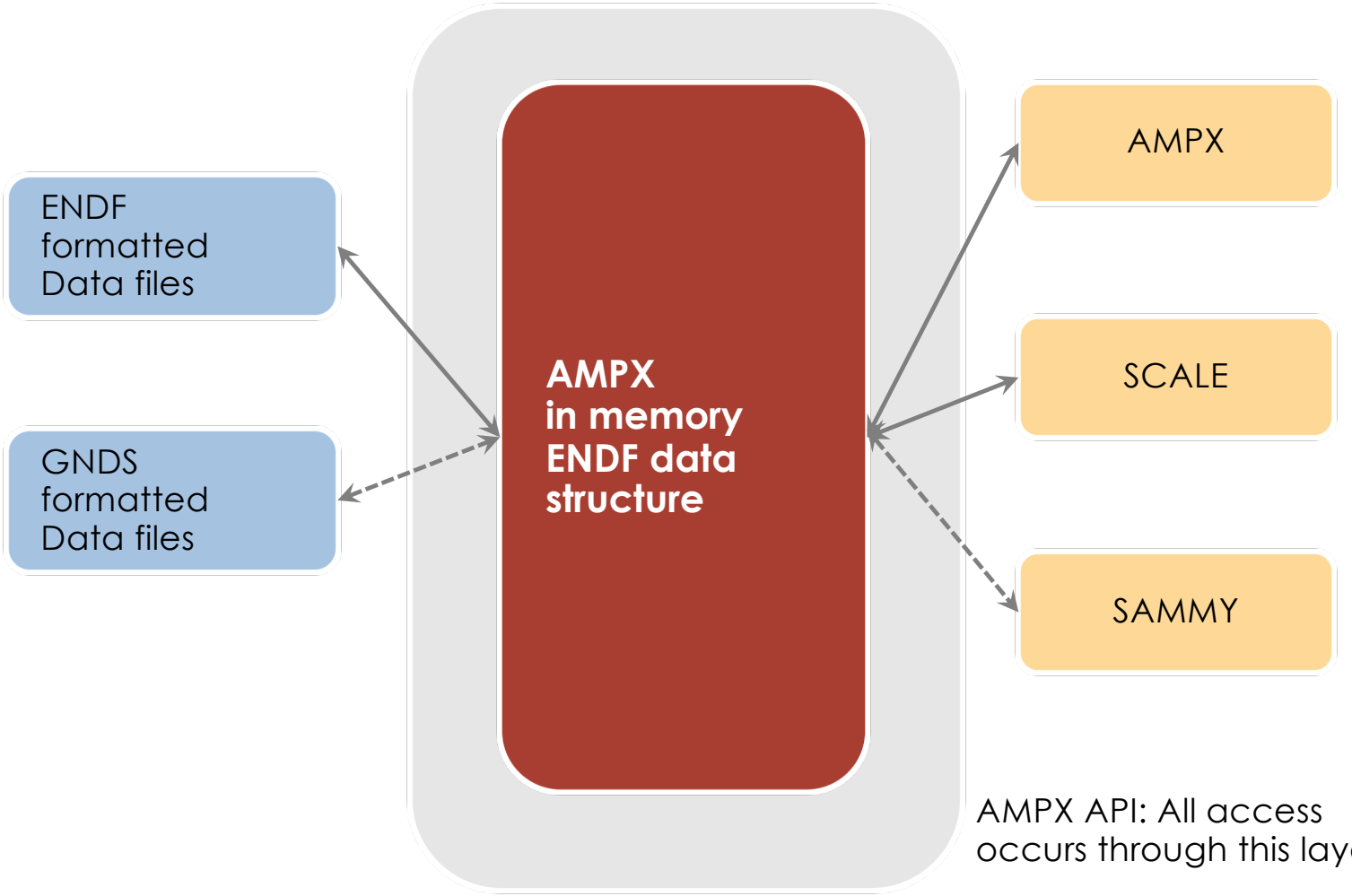
Dorothea Wiarda

Andrew Holcomb

WPEC May 13, 2020

Subgroup 43

U.S. DEPARTMENT OF ENERGY


ENDF
formatted
Data files
GNDS
formatted
Data files
AMPX
in memory
ENDF data
structure
AMPX
SCALE
SAMMY
AMPX API: All access
occurs through this layer

Original Implementation

- Hand-coded C++ classes that read GNDS classes and populate the AMPX in-memory structure.
- Supports 1-D data (including resonance reconstruction) and covariance information
- Available in the current SCALE beta version (recompile may be necessary)
- Tested by comparing processed results from GNDS formatted files to results from ENDF formatted files.

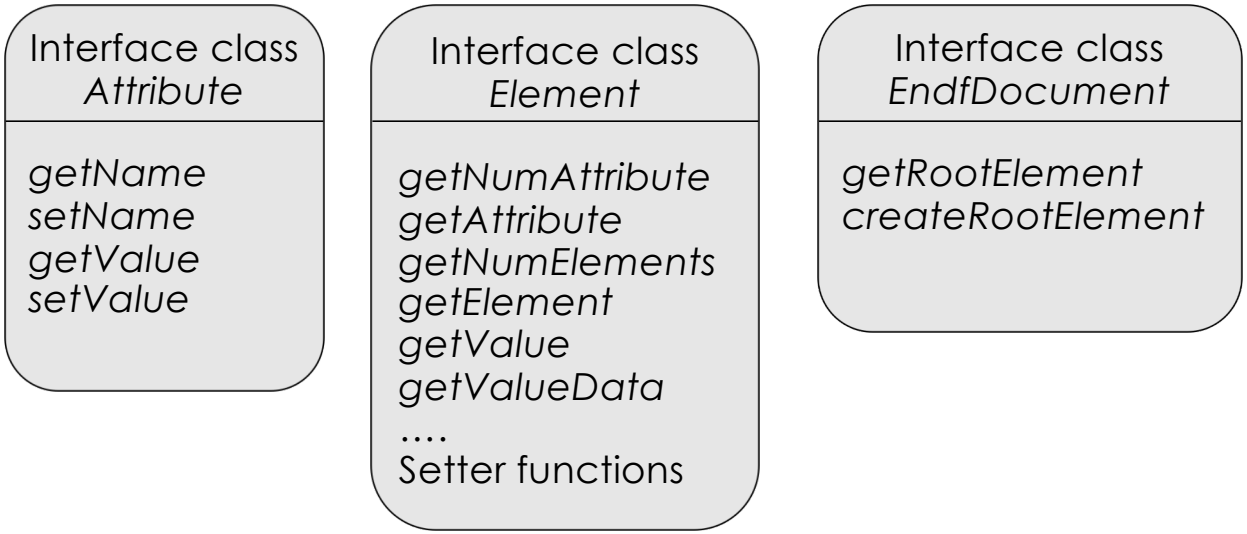
Now that GNDS is defined by JSON files, why not write code that reads JSON files and generates the low-level access classes.

Please note: This is not an API, just a layer on which the API can be build.

Therefore we will add the low-level access layer and from those classes fill the AMPX in-memory structure.

Some layers from the initial implementation are kept:

The direct access to the XML (or hd5 or JSON) GNDS file is abstracted into access to elements and attributes:



Currently using QT XML but will change to pugixml as SCALE has started to use it.

Some low-level containers are also reused

- Container: *name*, *index* and *label*
- Classes for GNDS elements: *array*, *value*, *table*, column, *columnHeader*, *data*, *externalFiles* and *externalFiles (*column, *columnHeader,* and *data* describe the content of the table element*)*
- GNDElement inherits from Container, contains a link to *externalFiles* (if applicable) and the Element object that describes the data. This is the base class.

All other low-level classes are written from JSON files

- Started with a C++ class that generated the low-level access classes in C++.
- Add the layer to fill the AMPX in-memory structure for 1-D, resonance data, 2-D (excluding S(alpha, beta)).
- Test by comparing processing from ENDF and GNDS formatted files
- Currently in the process of converting the initial C++ generator class to Python for easier use and maintainability.

Low-level access is not an API

In order to use the low-level access classes that are automatically generated to fill the AMPX in-memory structure, another layer is necessary:

- Utility functions to select the correct style of data desired – including inheritance of styles.
- More convenient accessor to the PoPs database in the context of AMPX.
- Functions that fill our resonance parameter objects to be described in terms of spin-groups, resonances etc. instead of elements in a table.
- Combine the various 1-D data to a "Tab1" object.
- Sort the kinematic data into the AMPX structures, again to convert low level access to names and functions that make sense to the user
- Utility functions that convert units to customary AMPX units

Implement AMPX in-memory setting

- Codes like POLIDENT and PUFF expect to get a file name from the user and then call our ENDF reading routines to fill the data they need.
- BaseGndReader is the implementation for GNDS for these function calls.

endfgnd::BaseGndReader Class Reference

```
#include <BaseGndReader.h>
```

Classes

class	**CovarianceInfo**

Public Member Functions

	BaseGndReader ()
	BaseGndReader (const **BaseGndReader** &orig)
virtual	**~BaseGndReader** ()
void	**setDataElements** (std::shared_ptr< const **endfgnd::ReactionSuite** > r, std::shared_ptr< const **endfgnd::CovarianceSuite** > c)
std::string	**getEvaluatedLabel** () const
void	**fillEvalutionInfo** (endf::EvaluationInfo &eval, const std::string &evalLabel="") const
void	**get1DPointWiseData** (endf::Tab1Container &tabCont, const std::string &evalLabel="") const
void	**getFissionData** (endf::FissionData &fissData, const std::string &evalLabel="") const
void	**getResonanceInfo** (endf::ResonanceInfo &resInfo, const std::string &evalLabel="", std::shared_ptr< const **CovarianceInfo** > covariance=NULL) const
void	**getCovarianceInfo** (endf::CovarianceContainer &covContainer, const std::string &evalLabel="") const
void	**getKinematicRawData** (**endfgnd::ampx::KinematicData** &kinContainer, double awp=-1, double zap=-1, const std::string &evalLabel="") const

Basic data types

Definitions

typedef TextTypes	**endfgnd::attributeValue**
typedef TextTypes	**endfgnd::bodyText**
typedef TextTypes	**endfgnd::UTF8Text**
typedef TextTypes	**endfgnd::printableText**
typedef TextTypes	**endfgnd::quotedText**
typedef TextTypes	**endfgnd::tdText**
typedef int	**endfgnd::Integer32**
typedef long int	**endfgnd::Integer64**
typedef unsigned int	**endfgnd::UInteger32**
typedef unsigned long int	**endfgnd::UInteger64**
typedef double	**endfgnd::Float64**
typedef float	**endfgnd::Float32**
typedef bool	**endfgnd::Boolean**
typedef std::runtime_error	**endfgnd::gnd_error**

Functions

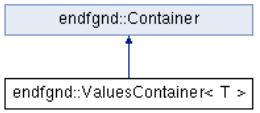
template<class T > void	**endfgnd::parseValue** (const std::string &val, T &want)
template<class T > void	**endfgnd::saveValue** (const T &val, std::string &text)
template<class T > T	**endfgnd::getDefaultValue** ()
template<class T > std::string	**endfgnd::getTypeName** (const T &v)
template<> void	**endfgnd::parseValue** (const std::string &val, bool &want)
template<> void	**endfgnd::parseValue** (const std::string &val, int &want)
template<> void	**endfgnd::parseValue** (const std::string &val, unsigned int &want)
template<> void	**endfgnd::parseValue** (const std::string &val, unsigned long &want)
template<> void	**endfgnd::parseValue** (const std::string &val, long &want)

ValuesContainer

endfgnd::ValuesContainer< T > Class Template Reference

`#include <ValuesContainer.h>`

Inheritance diagram for endfgnd::ValuesContainer< T >:



Public Member Functions

	ValuesContainer ()
	ValuesContainer (**endfgnd::Integer32** s, **endfgnd::Integer32** l, const std::string &type)
	ValuesContainer (const **ValuesContainer** &orig)
virtual	**~ValuesContainer** ()
virtual **endfgnd::Integer32**	**getStart** () const
virtual void	**setStart** (**endfgnd::Integer32** s)
virtual **endfgnd::Integer32**	**getLength** () const
virtual void	**setLength** (**endfgnd::Integer32** l)
virtual **endfgnd::Integer32**	**getSize** () const
virtual void	**setSize** (**endfgnd::Integer32** l)
virtual T	**getValue** (**endfgnd::Integer32** pos) const
virtual void	**setValue** (**endfgnd::Integer32** pos, T &v)
virtual std::string	**toString** (const std::string &indent) const
virtual void	**readFromElement** (std::shared_ptr< const **endfgnd::Element** > val, **ContainerRepository** *repo=NULL)
virtual void	**saveToElement** (std::shared_ptr< **endfgnd::Element** > val) const
virtual **Container** *	**getCopy** () const

▸ Public Member Functions inherited from **endfgnd::Container**

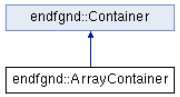
OAK RIDGE National Laboratory

ArrayContainer

endfgnd::ArrayContainer Class Reference

`#include <ArrayContainer.h>`

Inheritance diagram for endfgnd::ArrayContainer:



Public Types

enum	**COMPRESSION** { **NONE** = 0, **COMPRESSION::DIAGONAL** = 1, **COMPRESSION::FLATTENED** = 2, **COMPRESSION::EMBEDDED** = 3 }
enum	**PERMUTATION** { **NONE** = 0, **PERMUTATION::PLUS_1** = 1, **PERMUTATION::MINUS_1** = -1 }
enum	**STORAGE** { **STORAGE::ROW_MAJOR** = 0, **STORAGE::COLUMN_MAJOR** = 1 }
enum	**TRIANGULAR** { **NONE** = 0, **NONE** = 0, **NONE** = 0, **UPPER** = 1, **LOWER** = 2 }

Public Member Functions

	ArrayContainer ()
	ArrayContainer (const **ArrayContainer** &orig)
virtual	**~ArrayContainer** ()
virtual void	**readFromElement** (std::shared_ptr< const **endfgnd::Element** > val, **ContainerRepository** *repo=NULL)
virtual std::string	**toString** (const std::string &indent="") const
virtual void	**saveToElement** (std::shared_ptr< **endfgnd::Element** > val) const
virtual **endfgnd::UInteger32**	**getNumDim** () const
virtual **endfgnd::Integer32**	**getSize** (int dim) const
virtual void	**setShape** (std::shared_ptr< **endfgnd::ValuesContainer**< **endfgnd::UInteger32** > > s)
virtual bool	**checkValidContent** () const
virtual std::shared_ptr< const **endfgnd::ValuesContainer**< **endfgnd::UInteger32** > >	**getOffset** () const
template<class T >	
T	**getValue** (const std::vector< **endfgnd::Integer32** > indices) const
virtual **Container** *	**getCopy** () const

▸ **Public Member Functions inherited from endfgnd::Container**

OAK RIDGE National Laboratory

AMPX Release

- We hope to have the current implementation available in one of the upcoming SCALE beta releases.
- We have permission to distribute AMPX as open source (https://code.ornl.gov/RNSD/AMPX). But since AMPX is developed within SCALE we need to solve logistic problems to make sure only open source code gets distributed.
- Once those problems are solved, the current implementation will be available.
- We expect to use our API in SAMMY to save new resonance evaluations directly into GNDS format.