Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

History of the editions

Edition	Author	Comment
1.0	Fabrice MAUPIN	Creation
1.1	Fabrice MAUPIN	Remove “The sequence manager” section, revision of the “The integration step”, add new section “The limits of KEMET_API”
1.2	Fabrice MAUPIN	Update “approach by component” illustration, revision of “KEMET_DATA” section

Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

Table of Contents

About this guide....5
The limits of KEMET API....5
System, technical & fonctionnal requirements....5
Composition of ZIP file....6
The “Tests” directory....6
The “integration” directory....7
The “xml” directory....7
Technical Architecture....7
Functional notions....10
Transliteration to Hieroglyphs....10
Hieroglyphs to Transliteration....10
Manual for the encoding of Hieroglyph texts for Computer-Input....10
The KEMET_DATA directory....11
“phonograms.xml” file....11
“hiero.xml” file....11
“SupportedSequenceFormats.xml” file....12
“AssociatedProcToSFItems.xml” file....13
“lexicographical_list.xml” file....13
The “hieroglyphs” directory....13
The KEMET_LIB directory....13
Javadoc....14
Overview of Information System....14
The Localization of messages....15
How to configure KEMET API ?....16
How to use basic operations ?....18
How to extract particular phonograms ?....18
How to transliterate hieroglyph text from Gardiner's sign list ?....19
How to verify if the format of transliterated sequence is correct ?....20
How to get Gardiner's sign list from transliterated text ?....21
How to get phonetic read from transliterated text ?....22
Without global approach....22
With global approach....23

Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

How to epurate the phonetic complements (for biliteral and triliteral only) ?....23
The “integration” step....24
The “Manual for the encoding of Hieroglyph texts for Computer-Input”....25
The supported codes....25
How to initialize the location manager ?....25
Special caracters are they correct ?....26
how to use “ - ” and “ * ” codes ?....26
How to use “:” code ?....28
How to use the end of line ?....28
How to use the “ (“ and “) ” codes ?....30
Some examples....32
Glossary....32
GNU Free Documentation License....33

Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

Index of illustrations

Illustration 1: example of ZIP files....6
Illustration 2: approach by component....8
Illustration 3: Overview Information System of KEMET API....15

Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

About this guide

This guide has not for vocation to be a hieroglyphic grammatical reference.

It contents with explaining how to use the features offered by the KEMET API.

You will thus find piece of sources and explanations for every described feature.

You can modify this document if you consider it useful (under the terms of the GNU Free Documentation License).

In that case, please send your modifications at this electronic address: **adm.projetkemet@gmail.com**.

The limits of KEMET API

For the moment, it's not possible to transcribe the hieroglyphs, only to transliterate them. It's necessary to implement lexicographical lists for this feature.

It will be the object of future versions of KEMET API.

System, technical & fonctionnal requirements

At the system level, you have to install at least Java 6.

At the technical level, the good practice of the language JAVA is compulsory.

We advise you to read attentively the "READ_ME.txt" file delivered with this ZIP archive for KEMET API installation and use.

As regards the notions associated with the transliteration and the transcription of hieroglyphs, we advise you to consult the specialized books.

This document supplies only some hieroglyphic pristines used by the library and a glossary to explain the main terms.

Composition of ZIP file

You have to have this list of files in your ZIP file.

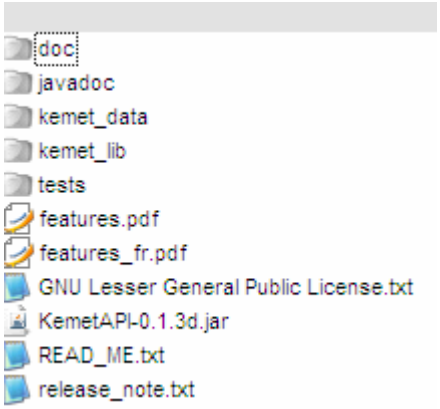

Illustration 1: example of ZIP files

We advise you to consult the "release_note.txt" file for more informations about news.

Important : the "features_x.pdf" files describes the hieroglyphic features which are implemented in this package.

If you downloaded the sources of this package, we will find all informations about (re)build this package. (please consult the "read_me.txt" file).

The "Tests" directory

Several tests are supply in the package. They are distributed in differents directories.

To run all the tests execute the "***AllTests.java***" test suite.

They are the guarantee of the robustness of the library.

The "integration" directory

It contains all tests about integration of hieroglyphics features.

We advise you to consult the list of supported features (PDF files supply with this package - "features.pdf" or "features_fr.pdf").

The "xml" directory

It contains all tests about integrity of data (hieroglyphs and phonograms).

Technical Architecture

Please find attached a synthesis of functionnal components of the KEMET API.

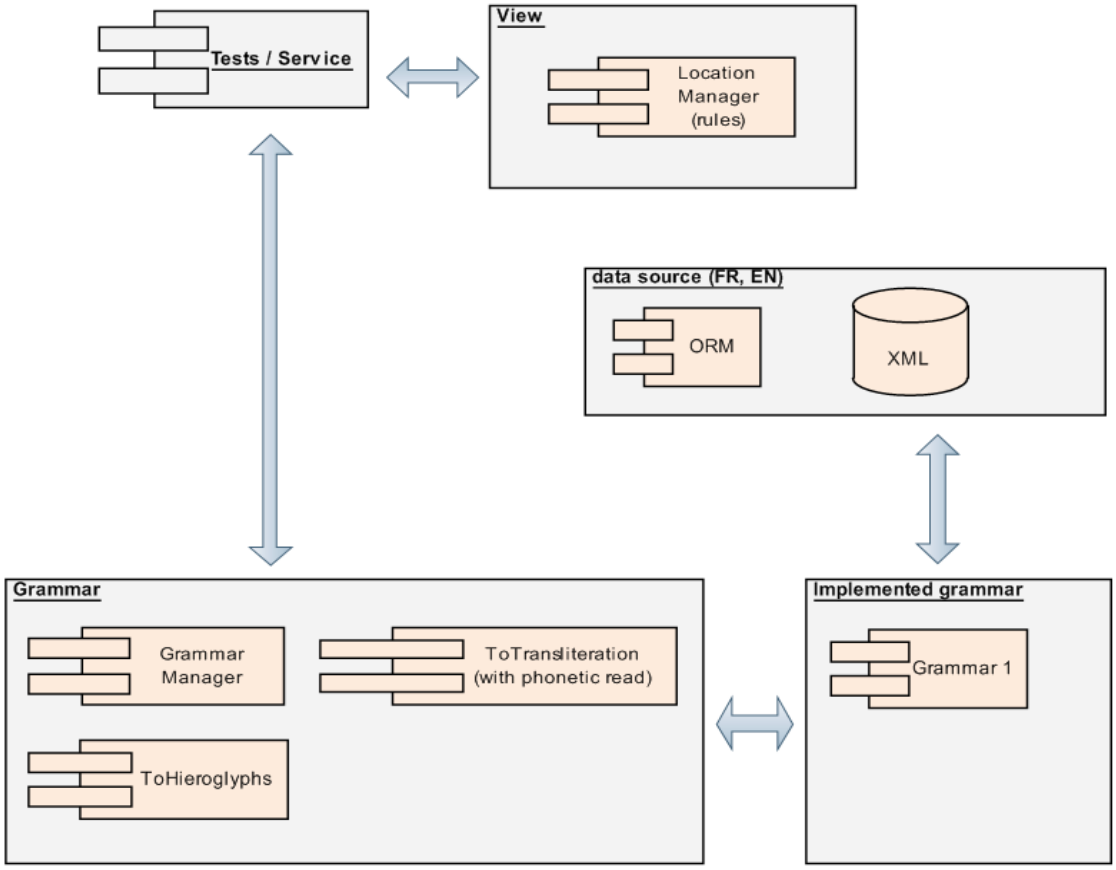

Illustration 2: approach by component

It's a approach by functionnal components.

Tests / Service : This layer allows to use the KEMET API (and these components). It can implement in the form of tests (Junit) or of sources traditional.

View : This layer doesn't yet implement a complete GUI layer. It supplies at first the means to position hieroglyphs in a "railing" consisted of lines and cadrats (Location manager).

Data source : This layer allows the access to the data contained in XML files.

It uses two approaches:

1) the use of the XPath technology
2) the use of an API (Castor XML) which allows to associate the data with objects in memory (mapping).

According to the speed with which we wish to reach the data, we shall choose the one or the other method.

Why to have chosen XML files ?

It is simple:

1) the volume of the data is not sufficient for the implementation of a database
2) to simplify the installation and the starting of the Library.

Grammar : This layer implements the grammar manager.

It plays the role of a "controler" between mainly the tests / service layer and the implemented grammar.

The name of the class as well as name of the package of the implemented grammar must be specified during the configuration of the KEMET API.

See “Step 1 : How to configure KEMET API ?” section.

These parameters are used by the grammar manager to execute methods of implemented grammar.

It crosses the demands of the tests / service layer for the implemented grammar which execute them.

It leans on generic features which are defined and executed in the implemented grammar (“ToHieroglyphs”,”ToTransliteration (with phonetic read)”).

This approach is going to allow terms to envisage the implementation of several grammars (At present a single grammar is implemented).

Implemented grammar : This layer implements all grammatical features which are defined.

Functional notions

Transliteration to Hieroglyphs

This functional notion has for objective the conversion of a sequence transliterated in hieroglyphic signs (Gardiner's codification and not the visual pictogram).

It can be used in a autonomous way.

Hieroglyphs to Transliteration

This functional notion has for objective the word for word reading of a hieroglyphic sequence and the conversion of this one in symbols appropriate for the transliteration.

It can be used in a autonomous way.

Manual for the encoding of Hieroglyph texts for Computer-Input

This manual has for objective the management of the graphic arrangement(measure,disposal) of hieroglyphs.

It leans on a normalized codification to reach this objective.

KEMET API implements some normalized codification of this manual.

It makes no control over the lexical validity of hieroglyphs (signs or transliterations) which must be diagrammatically arranged.

It can be used in a autonomous way.

See the javadoc to know what codifications are used.

The KEMET_DATA directory

See the javadoc for more information on the methods for how to use the differents files.

“phonograms.xml” file

This file contains some type of phonogram (uniliteral, biliteral, triliteral).

The “phonograms.xsd” file describes associated schema to “phonograms.xml” file.

Main attribute	Explanation
PhonogramType	Type of phonogram : 1->uniliteral, 2->biliteral, 3->triliteral
id	Id of sign (hieroglyph) if you have for example : “*<id>M17-M17</id>*” the “-” caracter means that this phonogram consists of the same sign repeated several times

Cf. “Overview of Information system” section.

“hiero.xml” file

This file contains some hieroglyphs based of Gardiner's List. It's available in French and in English.

The “hiero.xsd” file describes associated schema to “hiero.xml” file.

IMPORTANT : a special convention was adopted for coding some “hieroglyphic” caracters.

We didn't use “hieroglyph” font so it was necessary to use an other system to represent these caracters.

For example we find in the file : “h.”, “h)”, “s^”, ”h_”.

Our objective is to be independent for the moment from one font.

Main attribute	Explanation
id	Gardiner's codification
hieroglyph	Bitmap file path (in KEMET_DATA directory)
identification [optional]	Description of the hieroglyph
value	Transliterated value
type [optional]	1-> consonant, 2->semi - consonant
conventionnal_designation [optional]	Conventional designation
conventionnal_value [optional]	Pronunciation value
phonetic_value [optional]	Pronunciation value - phonetic side

For others attributes, please consult the "hiero.xsd" shema.

Cf. "Overview of Information system" section.

"SupportedSequenceFormats.xml" file

This file contains all supported formats of sequence.

Attribut	Explanation
format	Contains "uniliteral" or "biliteral" or "triliteral" keyword

The "+" separator allows to begin to group 2 keywords and to consider them as one all. The ";" separator allows to separate 2 keywords. A sequence can contain several keywords.

This file is used in purely directed purposes "test". If a format does not exist, the sequence manager indicates it (in log file). In that case it is necessary to add it in the file.

We thus have a useful list of formats to estimate what was already tested. It's very important to coverage tests.

This file is "connected" in the sequence manager.

The "SupportedSequenceFormats.xsd" file describes associated schema to "SupportedSequenceFormats .xml" file.

"AssociatedProcToSFItems.xml" file

This file describes all treatments for a particular notion to execute one after the other.

Attribut	Explanation
name	Name of the notion to treat
package	Name of the package who are implemented the treatments
className	Name of the class which contains all associated treatments
Method / name	Name of the method to treat

For example, it's necessary to "purify" phonetic complement for "biliteral" in a sequence.

For "purify" these phonetic complements, it requires certain number of treatments which are described in this file with "biliteral" as "name" attribute value.

See the javadoc for more information on the methods which uses it.

The "AssociatedProcToSFItems.xsd" file describes associated schema to "AssociatedProcToSFItems.xml" file.

"lexicographical_list.xml" file

This file contains currently a list of nouns. It is available in French and English.

The "lexicographical_list.xsd" file describes associated schema to "lexicographical_list .xml" file.

The "hieroglyphs" directory

This directory contains painted bitmap files (free of right) about hieroglyphs which are used in KEMET project.

The KEMET_LIB directory

This directory contains all librairies which are useful to use KEMET API.

Kemet Project

Developer Guide for Kemet API (0.2)

v0.2.3d

Library	Explanation
castor	Useful to map relational database or file to objects in memory
commons-logging	Used by castor
Log4j (*)	Useful for KEMET API to log trace

See "READ_ME.txt" file to know how to install these librairies.

(*) In "src" directory, there is a "log4j.properties" file which can be modified according to your needs.

Javadoc

It supplies only the public part of the Library. Only some methods are described in this document.
We advise you to consult in details these methods.

Overview of Information System

Please find attached a synthesis of the information system of the KEMET API.

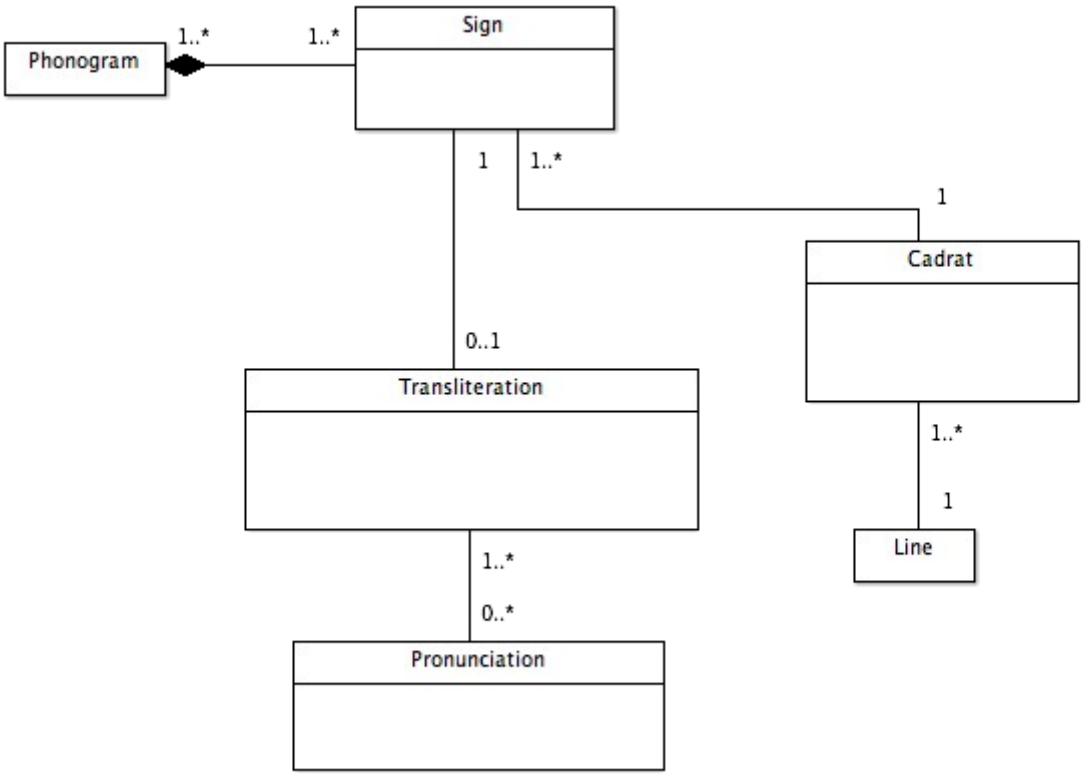

Illustration 3: Overview Information System of KEMET API

For more information, please consult the XSD files which are in the KEMET_DATA directory.

The Localization of messages

The localization for messages is automatically managed .

The list of supported localizations :

– "fr, "FR"

– "en", "US".

We take into account by default the localization local file.

If the localization local file doesn't exist, the "US" localization is used.

See the javadoc for more informations about the use of Messages class.

How to configure KEMET API ?

It is necessary to initialize the grammar manager.

To do that, this initialization is exclusively made in the source. You have to initialize at first certain attributes.

Complete list of attributes

Attribute	explanation
grammarMainClassName	This attribute always has to have this value : "GrammarBusiness" (Cf. "Technical Architecture" section)
selectedGrammar	This attribute always has to have this value : "org.kemet.impl.grammar1" (Cf. "Technical Architecture" section)
XMLPath	KEMET_DATA directory path (local path where KEMET_DATA is installed) which contains all required data for KEMET API pay attention to unix or windows system for path (*)
file1	This attribute always has to have this value : Phonogram.xml (Cf. "The KEMET_DATA directory" section)
file2	This attribute always has to have this value : hiero.xml (Cf. "The KEMET_DATA directory" section)
encoding	This attribute always has to have this value : "UTF-8" ; All sources and files are encoded "UTF-8" there.
SupportedSequenceFormats [optional]	This attribute always has to have this value : SupportedSequenceFormats.xml (Cf. "The KEMET_DATA directory" section)
SequenceFormatManager [optional]	This attribute always has to have this value : AssociatedProcToSFItems.xml (Cf. "The KEMET_DATA directory" section)
methodNameToManageTransliteration	Method name (without signature) to process hieroglyphs to transliteration (Cf. "Technical Architecture" section)

Attribute	explanation
methodNameToManageHieroglyphs	Method name (without signature) to process transliteration to hieroglyphs (Cf. "Technical Architecture" section)
methodNameToPhoneticRead	Method name (without signature) to process phonetic read (Cf. "Technical Architecture" section)

(*) for unix system : \\Volumes\\... | for windows system : C..Z:\\...

Certain attributes are optional. Please consult the javadoc to know if a feature needs an attribute in particular.

In the example which follows all attributes are initialized.

Source example (on Windows system) :

```
private GrammarManager gm;
public static final String grammarMainClassName = "GrammarBusiness";

public static final String selectedGrammar = "org.kemet.impl.grammar1";

private HashMap args;

public static String XMLPath = "D:\\DEV\\Kemet\\workspace\\kemet_data\\";

public static String file1 = XMLPath + "phonogram.xml";
public static String file2 = XMLPath + "hiero.xml";

public static String SupportedSequenceFormats =
"D:\\DEV\\Kemet\\workspace\\kemet_data\\SupportedSequenceFormats.xml";

public static String SequenceFormatManager =
"D:\\DEV\\Kemet\\workspace\\kemet_data\\AssociatedProcToSFItems.xml";

// init the param...
args = new HashMap();

args.put("XMLPath", XMLPath);
args.put("file1", file1);
args.put("file2", file2);
args.put("encoding", "UTF-8");
```

```
args.put("SupportedSequenceFormats", SupportedSequenceFormats);
args.put("SequenceFormatManager", SequenceFormatManager);

args.put("methodNameToManageTransliteration",
                        “toTransliteration”);
args.put("methodNameToManageHieroglyphs",“toHieroglyphs”);
args.put("methodNameToPhoneticRead", “toPhoneticRead”);

// init the grammar manager...
gm = new GrammarManager(selectedGrammar + "." + grammarMainClassName,
                                        args);
```

How to use basic operations ?

How to extract particular phonograms ?

You have to initialize the grammar manager (see “First Step : How to configure KEMET API ?” section).

Source example :

```
private Hashtable<String, String> phInfo;

// get all phonograms with some informations (sign, type)
phInfo = org.kemet.impl.grammar.model.Phonogram.getAllPhonograms();
```

See the javadoc for more information.

The grammar manager loads all phonograms by default.

You can choose to work with a type of phonogram in particular.

Source example :

```
// list of available values
public static final int uniliteral = 1;

public static final int biliteral = 2;

public static final int triliteral = 3;
```

```
		public static final int ALL = 4; // all type of phonograms

		// get the uniliterals phonograms
		org.kemet.impl.grammar.model.Phonogram.getPhonogramsByType(uniliteral);

		// or
		// Phonogram[] ph =
// org.kemet.impl.grammar.model.Phonogram.getPhonogramsByType(uniliteral);
```

You can set the return value but it's not compulsory.

By default, KEMET API stored in private static attribut named "extractedPhonograms" the return value.

This value is used by others treatments for more ease.

See the javadoc for more information.

How to transliterate hieroglyph text from Gardiner's sign list ?

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

You have to use Gardiner's codification.

You pass a array of "id" (with "_" between 2 "id") and the method returns a string which contains transliterated "id" list.

There is no separator in the returned string.

Source example :

```
		// init the data tests...
		signList = new String[] { "G1_N29_G370", "G43_D58_N35_N80" };
		results = "3q?jwbn";

		// test if we have "???" expression
		try {
			assertEquals(gm.toTransliteration(new String[] { "???" }, 0), "");
		} catch (Exception e) {
			assertTrue(false);
		}
```

```
		// test if we have a right transliteration
		try {
			assertEquals(gm.toTransliteration(signList, 0), results);
		} catch (Exception e) {
			assertTrue(false);
		}
```

The "?" expression means that the sign can't be transliterate.

See the javadoc for more information.

How to verify if the format of transliterated sequence is correct ?

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

You have to use Gardiner's codification and pass a array of "id" (with "_" between 2 "id").

Source example :

```
		// init the data tests...
		signList = new String[] { "G1_N29_G370", "G43_D58_N35_N80" };

		/*
		 * (exclude (y/n) "?" caracter)
		 */
		try {
			assertEquals(gm.isTCorrectFormat(gm.toTransliteration(signList, 0),
					true), true);
		} catch (Exception e) {
			assertTrue(false);
		}

		try {
			assertEquals(gm.isTCorrectFormat(gm.toTransliteration(signList, 0),
					false), false);
		} catch (Exception e) {
			assertTrue(false);
		}

		assertEquals(gm.isTCorrectFormat("3x?", true), false);
```

See the javadoc for more information.

How to get Gardiner's sign list from transliterated text ?

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

Source example :

```
// init data tests...
transliterationList = new String[] { "jqr", "psh." };
results = "M17N29D21Q3[O34;S29]V28";

// the transliteration is not correct
try {
        assertEquals(gm.toHieroglyphs(new String[] { "???" }), "");
} catch (Exception e) {
        assertTrue(false);
}

try {
        assertEquals(gm.toHieroglyphs(new String[] { "" }), "");
} catch (Exception e) {
        assertTrue(false);
}

// the transliteration is correct
try {
        assertEquals(gm.toHieroglyphs(transliterationList), results);
} catch (Exception e) {
        assertTrue(false);
}

// the transliteration is partially correct
try {
        assertEquals(gm.toHieroglyphs(new String[] { "jer" }), "M17D21");
} catch (Exception e) {
        assertTrue(false);
}

try {
        assertEquals(gm.toHieroglyphs(new String[] { "j.r" }), "M17D21");
} catch (Exception e) {
        assertTrue(false);
```

```
}

try {
        assertEquals(gm.toHieroglyphs(new String[] { "?jer" }), "M17D21");
} catch (Exception e) {
        assertTrue(false);
}
```

The values between hooks represent "id" which have the same transliterated value. They are separated by ";".

In this example for the transliteration which are partially correct, you note that the method is able to get only correct transliterated value.

See the javadoc for more information.

How to get phonetic read from transliterated text ?

Without global approach

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

This approach consists in making a phonetic reading of each letter of each transliteration.

Source example :

```
// init data tests...
transliterationList = new String[] { "jqr", "d_d" };
results = "ikrdjd";

transliterationList2 = new String[] { "s", "3tp" };
results2 = "s (deaf)éatèp";

// phonetic read without vowels
try {
        assertEquals(gm.toPhoneticRead(transliterationList, false),
                        results);
} catch (Exception e) {
        assertTrue(false);
}

// phonetic read with vowels and complementary information
```

```
                    try {
                            assertEquals(gm.toPhoneticRead(transliterationList2, true),
                                                    results2);
                    } catch (Exception e) {
                            assertTrue(false);
                    }
```

See the javadoc for more information.

With global approach

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

This approach consists in making a phonetic reading of each transliteration.

Source example :

```
            // tests exceptions to pronunciation particulars cases
            try {
                    assertEquals(gm.toPhoneticReadWithGlobalApproach(new String[] {
                                            "mmj", "mj" }, false), "mam");
            } catch (Exception e) {
                    assertTrue(false);
            }

            // phonetic read with vowels
            try {
                    assertEquals(gm.toPhoneticReadWithGlobalApproach(new String[] {
                                            "mmj", "mj" }, true), "méame");
            } catch (Exception e) {
                    assertTrue(false);
            }
```

See the javadoc for more information.

How to epurate the phonetic complements (for biliteral and triliteral only) ?

You have to initialize the grammar manager (see "First Step : How to configure KEMET API ?" section).

Source example :

```
// init tests data...
d1 = new String[] { "bb33", "kmm" };
d1Expected = new String[] { "b3", "km" };

private Hashtable<String, String> phInfo;

// get all phonograms with some informations (sign, type)
phInfo = org.kemet.impl.grammar.model.Phonogram.getAllPhonograms();

// treat phonetic complements
assertEquals(Arrays.equals(gm.toEpuratePhoneticComplementFromSequence(
                        d1, 2, phInfo, args), d1Expected), true);
```

See the javadoc for more information.

The “integration” step

The “integration” step is the higher level of this package.

This step is very important : it allows to test all the implemented hieroglyphic features (we'll find the list of the features in “features_xx.pdf” files).

2 “toolbox” are supply : the hieroglyph manager and the transliteration manager.

The Hieroglyph manager allows :

- to convert the id of the hieroglyphs to a transliterated sequence
- to convert the id of the hieroglyphs to a list of types (ex : UNILITERAL, PRONOUN)
- to convert the id of the hieroglyphs to a phonetic read.

The transliteration manager allows :

- to “cut” the transliterated sequence in “small” parts (ex : “jwnn” devient “jwn”,”n”)
- to convert the transliterated sequence to a list of hieroglyphs
- to convert the transliterated sequence to a list of types.

For more informations, please consult the tests in this package.

The “Manual for the encoding of Hieroglyph texts for Computer-Input”

This manual explain how to organize the view of hieroglyphs.

We use “Line” and “Cadrat” (part of a line) to represent hieroglyphs.

We advise you of consult this manual (for more information see “READ_ME.txt” file which contains the name of publisher).

The supported codes

Please find attached the list of supported codes.

Source example :

```
HashMap values = new HashMap();

// Supported MdC code
values.put("-", "ruleIndentVerify");
values.put("*", "ruleStarVerify");
values.put(":", "ruleDoublePointVerify");
values.put("!", "ruleExclamationPointVerify");
values.put("(", "ruleInputBracketVerify");
values.put(")", "ruleOutputBracketVerify");
```

How to initialize the location manager ?

You have to initialize the supported codes (see “The supported codes” section).

Source example :

```
private String input = "N1-N2-G1";
 LocationManager ra = null;

try {
        ra = new LocationManager(input, “org.kemet.mdc.LocationRules”, values);
```

```
        } catch (LocationManagerException e) {
                assertTrue(true);
        }
```

Special caracters are they correct ?

You have to initialize the supported codes (see "The supported codes" section).

Source example :

```
        LocationManager ra = null;
        private String input = "N1*N2!G1";

        try {
                ra = new LocationManager(input, "org.kemet.mdc.LocationRules", values);
                assertEquals(ra.getCorrect().getValue(), EReturnedMdCCode.Ok
                                                        .getValue());
        } catch (LocationManagerException e) {
        }
```

List of EReturnedMdcCode values = NoSpecialCaracters(0), Ok(1), NotOk(2);

*how to use " - " and " * " codes ?*

Source example :

```
        // 1. "-"
        private String input = "N1-N2-G1";

        LocationManager ra = new LocationManager();
        ra.getInformationLocation(input);

        assertEquals(Line.size(), 1);

        assertEquals(Line.get(0).getCadratList().size(), 3);

        assertEquals(Cadrat.get(0).getObject(0).getJustification(),
                                EJustification.CENTER.getValue());

        assertEquals(Cadrat.get(0).getObject(0).getIndex(), 1);
```

```
// 2. "*"
input = "N1*N2*G1";
ra.getInformationLocation(input);

assertEquals(Line.size(), 1);

assertEquals(Line.get(0).getCadratList().size(), 1);
assertEquals(Cadrat.get(0).getObject(0).getJustification(),
                        EJustification.LEFT.getValue());

// 3. "*", "-"
input = "N1*N2-G1";
ra.getInformationLocation(input);

assertEquals(Line.size(), 1);
assertEquals(Line.get(0).getCadratList().size(), 2);

assertEquals(Cadrat.get(0).getObject(0).getJustification(),
            EJustification.LEFT.getValue());

assertEquals(Cadrat.get(0).getObject(1).getJustification(),
            EJustification.RIGHT.getValue());

assertEquals(Cadrat.get(1).getObject(0).getJustification(),
            EJustification.CENTER.getValue());

// 4. tests for justification
input = "N1*N2*X1*A15-G1*N3-G1-N2";
ra.getInformationLocation(input);

assertEquals(Line.size(), 1);
assertEquals(Line.get(0).getCadratList().size(), 4);

assertEquals(Cadrat.get(0).getObject(0).getJustification(),
                        EJustification.NoJustification.getValue());
assertEquals(Cadrat.get(0).getObject(0).getIndex(), 1);

assertEquals(Cadrat.get(0).getObject(1).getJustification(),
                        EJustification.NoJustification.getValue());
assertEquals(Cadrat.get(0).getObject(1).getIndex(), 2);

assertEquals(Cadrat.get(1).getObject(0).getJustification(),
                        EJustification.LEFT.getValue());

assertEquals(Cadrat.get(2).getObject(0).getJustification(),
```

```
                                EJustification.CENTER.getValue());
        assertEquals(Cadrat.get(3).getObject(0).getJustification(),
                                EJustification.CENTER.getValue());

        // 5. tests for associated signs
        assertEquals(Cadrat.get(0).getSign(0), "N1");
        assertEquals(Cadrat.get(0).getSign(1), "N2");
        assertEquals(Cadrat.get(1).getSign(0), "G1");
```

See the javadoc for more informations.

How to use ":" code ?

Source example :

```
        LocationManager ra = new LocationManager();
        input = "N1*P1:N2";

        ra.getInformationLocation(input);

        assertEquals(Line.size(), 1);
        assertEquals(Line.get(0).getCadratList().size(), 2);

        assertEquals(Cadrat.get(0).getType(), ECadratType.upHalf.getValue());
        assertEquals(Cadrat.get(1).getType(), ECadratType.downHalf.getValue());

        ra = new LocationManager();
        input = "N1:P1-G1";
        ra.getInformationLocation(input);

        assertEquals(Line.size(), 1);
        assertEquals(Line.get(0).getCadratList().size(), 3);

        assertEquals(Cadrat.get(0).getType(), ECadratType.upHalf.getValue());
        assertEquals(Cadrat.get(1).getType(), ECadratType.downHalf.getValue());

        assertEquals(Cadrat.get(2).getType(), ECadratType.full.getValue());
```

See the javadoc for more informations.

How to use the end of line ?

Source example :

```
LocationManager ra = null;
private String[] input = new String[] {
                        "N1*P1:N2!N1:P1-G1", "N1*P1:N2!N1:P1-G1!" };

try {
            ra = new LocationManager(input[0], Messages
                                                .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
}

try {
            ra = new LocationManager(input[1], Messages
                                        .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
            assertTrue(true);
}

ra = new LocationManager();
ra.getInformationLocation(input[0]);

assertEquals(Line.size(), 2);

Cadrat c = (Cadrat) Line.get(0).getCadratList().get(0);
assertEquals(c.getType(), ECadratType.upHalf.getValue());

c = (Cadrat) Line.get(0).getCadratList().get(1);
assertEquals(c.getType(), ECadratType.downHalf.getValue());

c = (Cadrat) Line.get(1).getCadratList().get(2);
assertEquals(c.getType(), ECadratType.full.getValue());

c = (Cadrat) Line.get(1).getCadratList().get(0);
assertEquals(c.getObject(0).getJustification(), EJustification.CENTER
                                        .getValue());

c = (Cadrat) Line.get(0).getCadratList().get(0);
assertEquals(c.getObject(0).getJustification(), EJustification.LEFT
                                        .getValue());

c = (Cadrat) Line.get(0).getCadratList().get(0);
assertEquals(c.getObject(1).getJustification(), EJustification.RIGHT
                                        .getValue());
```

```
c = (Cadrat) Line.get(0).getCadratList().get(1);
assertEquals(c.getObject(0).getJustification(), EJustification.CENTER
                                        .getValue());
```

See the javadoc for more informations.

How to use the " (" and ") " codes ?

Source example :

```
private String[] input = new String[] { "G1*(N1:N2):P1", "G1*(N1:N2:P1",
                        "G1*)N1:N2:P1", "G1*((N1:N2)):P1" };
LocationManager ra = null;

try {
        ra = new LocationManager(input[0], Messages
                                        .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
}

try {
        ra = new LocationManager(input[1], Messages
                                        .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
        assertTrue(true);
}

try {
        ra = new LocationManager(input[2], Messages
                                    .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
                    assertTrue(true);
}

try {
        ra = new LocationManager(input[3], Messages
                                        .getString("LocationManager.4"), values);
} catch (LocationManagerException e) {
                    assertTrue(true);
}

ra = new LocationManager();
ra.getInformationLocation(input[0]);
```

```
assertEquals(Line.size(), 1);
assertEquals(Line.get(0).getCadratList().size(), 5);

Cadrat c = (Cadrat) Line.get(0).getCadratList().get(0);
assertEquals(c.getType(), ECadratType.full.getValue());

c = (Cadrat) Line.get(0).getCadratList().get(1);
assertEquals(c.getType(), ECadratType.rightquarter.getValue());

c = (Cadrat) Line.get(0).getCadratList().get(2);
assertEquals(c.getType(), ECadratType.upHalf.getValue());

c = (Cadrat) Line.get(0).getCadratList().get(3);
assertEquals(c.getType(), ECadratType.downHalf.getValue());

c = (Cadrat) Line.get(0).getCadratList().get(4);
assertEquals(c.getType(), ECadratType.downHalf.getValue());

// justification tests

c = (Cadrat) Line.get(0).getCadratList().get(0);
assertEquals(c.getObject(0).getJustification(), EJustification.CENTER
                                        .getValue());

c = (Cadrat) Line.get(0).getCadratList().get(1);
assertEquals(c.size(), 0);

c = (Cadrat) Line.get(0).getCadratList().get(2);
assertEquals(c.getObject(0).getJustification(), EJustification.CENTER
                                        .getValue());

c = (Cadrat) Line.get(0).getCadratList().get(4);
assertEquals(c.getObject(0).getJustification(), EJustification.CENTER
                                        .getValue());
```

See the javadoc for more informations.

Some examples

Please consult the "tests" directory supplies. It contains a suite of tests based of hieroglyphic features which were described.

Glossary

Please find attached the definition of the notions which are used by the KEMET API.

Notion	Definition
Transliteration	Word for word reading of a hieroglyphic sequence (with symbols appropriate for the transliteration)
Phonogram	Represent one or several sounds associated with a phonetic value
uniliteral	Phonogram containing that a single sound
biliteral	Phonogram containing 2 sounds
triliteral	Phonogram containing 3 sounds
Phonetic complement	Useful to differentiate 2 terms written in the same way - a phonetic complement does not pronounce
Semi - consonant	Can take the value of a consonant or a vowel

GNU Free Documentation License

GNU Free Documentation License
Version 1.3, 3 November 2008

0. PREAMBLE
The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

This License is a kind of "copyleft", which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS
This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document", below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you". You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related

matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The "Invariant Sections" are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The "Cover Texts" are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A "Transparent" copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not "Transparent" is called "Opaque".

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The "Title Page" means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, "Title Page" means the text near the most prominent appearance of the work's title, preceding the beginning of the body of the text.

The "publisher" means any person or entity that distributes copies of the Document to the public.

A section "Entitled XYZ" means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as "Acknowledgements", "Dedications", "Endorsements", or "History".) To "Preserve the Title" of such a section when you modify the Document means that it remains a section "Entitled XYZ" according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may

have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING
You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY
If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS
You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from

those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History", Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications", Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements". Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.
If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements", provided it contains nothing but endorsements of your Modified Version by various parties—for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-

Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS
You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements", and any sections Entitled "Dedications". You must delete all sections Entitled "Endorsements".

6. COLLECTIONS OF DOCUMENTS
You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS
A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the

Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION
Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

If a section in the Document is Entitled "Acknowledgements", "Dedications", or "History", the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION
You may not copy, modify, sublicense, or distribute the Document except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense, or distribute it is void, and will automatically terminate your rights under this License.

However, if you cease all violation of this License, then your license from a particular copyright holder is reinstated (a) provisionally, unless and until the copyright holder explicitly and finally terminates your license, and (b) permanently, if the copyright holder fails to notify you of the violation by some reasonable means prior to 60 days after the cessation.

Moreover, your license from a particular copyright holder is reinstated permanently if the copyright holder notifies you of the violation by some reasonable means, this is the first time you have received notice of violation of this License (for any work) from that copyright holder, and you cure the violation prior to 30 days after your receipt of the notice.

Termination of your rights under this section does not terminate the licenses of parties who have received copies or rights from you under this License. If your rights have been terminated and not permanently reinstated, receipt of a copy of some or all of the same material does not give you any rights to use it.

10. FUTURE REVISIONS OF THIS LICENSE
The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software

Foundation. If the Document specifies that a proxy can decide which future versions of this License can be used, that proxy's public statement of acceptance of a version permanently authorizes you to choose that version for the Document.

11. RELICENSING
"Massive Multiauthor Collaboration Site" (or "MMC Site") means any World Wide Web server that publishes copyrightable works and also provides prominent facilities for anybody to edit those works. A public wiki that anybody can edit is an example of such a server. A "Massive Multiauthor Collaboration" (or "MMC") contained in the site means any set of copyrightable works thus published on the MMC site.

"CC-BY-SA" means the Creative Commons Attribution-Share Alike 3.0 license published by Creative Commons Corporation, a not-for-profit corporation with a principal place of business in San Francisco, California, as well as future copyleft versions of that license published by that same organization.

"Incorporate" means to publish or republish a Document, in whole or in part, as part of another Document.

An MMC is "eligible for relicensing" if it is licensed under this License, and if all works that were first published under this License somewhere other than this MMC, and subsequently incorporated in whole or in part into the MMC, (1) had no cover texts or invariant sections, and (2) were thus incorporated prior to November 1, 2008.

The operator of an MMC Site may republish an MMC contained in the site under CC-BY-SA on the same site at any time before August 1, 2009, provided the MMC is eligible for relicensing.