YaPPI – Yet another Particle Property Interface

A Particle Property Database

Mark Dönszelmann[1], Patrick Hellwig[1], Tony Johnson[2], Paolo Palazzi[1], Mario Ruggier[1]

[1] *CERN, CH-1211 Geneva 23, Switzerland*
[2] *SLAC, P.O.Box 4349, Stanford CA 94309, California, U.S.A.*

Abstract

YaPPI[1] is an XML database of particle properties, accessible through an API, currently only for Java but other languages or bindings may be easily added. For portability and flexibility, the data is stored in XML. The goal is to have particle property data available programmatically in a way that is independent of the physicist's platform, environment, or analysis package being used.

Currently, the applications that use YaPPI via its Java API are (a) Java Analysis Studio (JAS)[2], a High Energy Physics analysis software package, where users may access particle properties in XML directly from within JAS, and (b) a Java servlet that allows online web browsing of particle properties.

The data is imported from the Review of Particle Physics (RPP) of the Particle Data Group (PDG)[3].

Keywords: Particle Properties; PDG; Java; XML; database.

1. Introduction

The goal of YaPPI is to make a small part of the particle data book available via an Application Programming Interface, or API. The YaPPI API has been implemented for the Java language. Adding bindings for other languages is easy, and will not involve any extra learning of other APIs.

The standard source for particle properties is the Particle Data Group [PDG]. For an experiment or analysis, a physicist would normally manually look up data needed for an application out of the Review of Particle Physics [RPP] book. Most physicists would simply like to have this information available in the analysis software. If there would be an official published database, the physicist could save the time-consuming and error-prone process of keying-in around 1000 different particles by hand, all with their properties and decay modes, for each separate application. Result is that each analysis software package must implement its own database, either hard-coded into its code sources or as a database system particular to the software package. This is a problem as,

1. URL: http://yappi.freehep.org/
2. URL: http://jas.freehep.org/
3. URL: http://pdg.lbl.gov/

for each case of new or modified data, each software package's administrator has to make sure to update their version of the database by hand. This situation is not exactly convenient.

YaPPI addresses this problem by building up a generic database with particle properties, accessible through a programmatic API. For genericity, platform and database independence, the data format used is XML [XML].

An interface for the Java Analysis Studio JAS [JAS] has so far been implemented, this being built on a lower level common API for Java. To minimize problems with data updates, we have made tools to convert data to XML, from the PDG MC format [PDG-MC], as well as from PDG produced PostScript.

2. XML

The Extensible Markup Language is the universal format for structured documents and data on the Web[4]. It developed out of SGML[5], which was implemented in the early 1980's (ISO standard since 1986). The W3C[6] recommendation for XML 1.0, was published in February 1998 [XML98]. XML allows the structuring of data in platform independent text files. Data is described by the use of opening and closing *tags* (logical labels, delineated with '<' and '>') and *attributes* (of the form *name*="*value*"), just like the contents of an HTML file. However, different to HTML, the function of a tag or attribute in XML is not predefined. XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data to the application that reads it.

The XML standard is the base of a family of associated technologies, each addressing a particular application domain. Examples would be XLink (description of hyperlinks), XPointer (identification of resource fragments), XSL & XSLT (for data rendering and transformations).

3. Architecture

The main task was to provide Java Analysis Studio (JAS) with particle properties. The choice of XML as database format for YaPPI fits well with the platform-independence of JAS. Another benefit of using XML is the wide range of available tools. There are XML parsers, to analyze and read XML, for almost every programming language. For this project we took the Apache Xerces XML parser for Java [XERCES].

4. See also "XML in 10 points" from Bert Bos, http://www.w3.org/XML/1999/XML-in-10-points
5. Standard Generalized Markup Language
6. World Wide Web Consortium: http://www.w3.org/

The data model

The YaPPI data model is shown in Figure 1. Apart from the basic requirement of fitting existing particle data, YaPPI's data model should also be able to handle storage of new particles, properties and other information in the database. To make sure we will be able to do this, we made the following general assumptions:

1. A particle always has a unique ASCII name (which serves as an index).
2. The particle name can optionally be expressed in LaTeX.
3. All Properties exist independently, thus the data entries can be stored separately without requiring cross references between them.
4. Particles are all stored by Name, and contain no information about order or hierarchy with respect to other particles. However, particle hierarchical information is stored in a separate family data structure that can have Particles as leaf nodes.
5. Decay modes are also stored independently of particle properties.

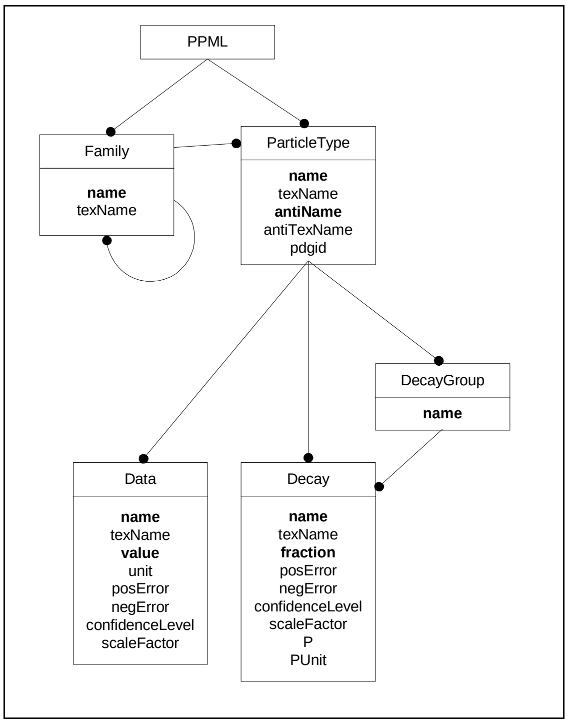


Figure 1: The YaPPI XML data model, showing element tags and their attributes. Required attributes are shown in bold. A ParticleType, in addition to its attributes, can contain arbitrary Data elements. Decays are stored in separate Decay elements, and may also be grouped in DecayGroups. Independently of particle properties, a hierarchy of Family elements (referring to particles by their name) defines the particle family tree.

Data flow

Three data levels can be categorized as (a) the master sources maintained by the PDG (b) the XML storage files and (c) any software application using these XML data. How the data moves around these three layers is indicated in Figure 2. First the data is converted from the PDG sources (currently directly to XML). The information in XML can then be read and provided to different applications as required, through the API. Currently, the following applications exist:

1. A common Java API, which allows Java Applications to use the data.
2. The Java Analysis Studio (JAS) Interface, a subset of the common Java API.
3. A Java Servlet [Servlet], to allow online browsing of data, (see Figure 6).

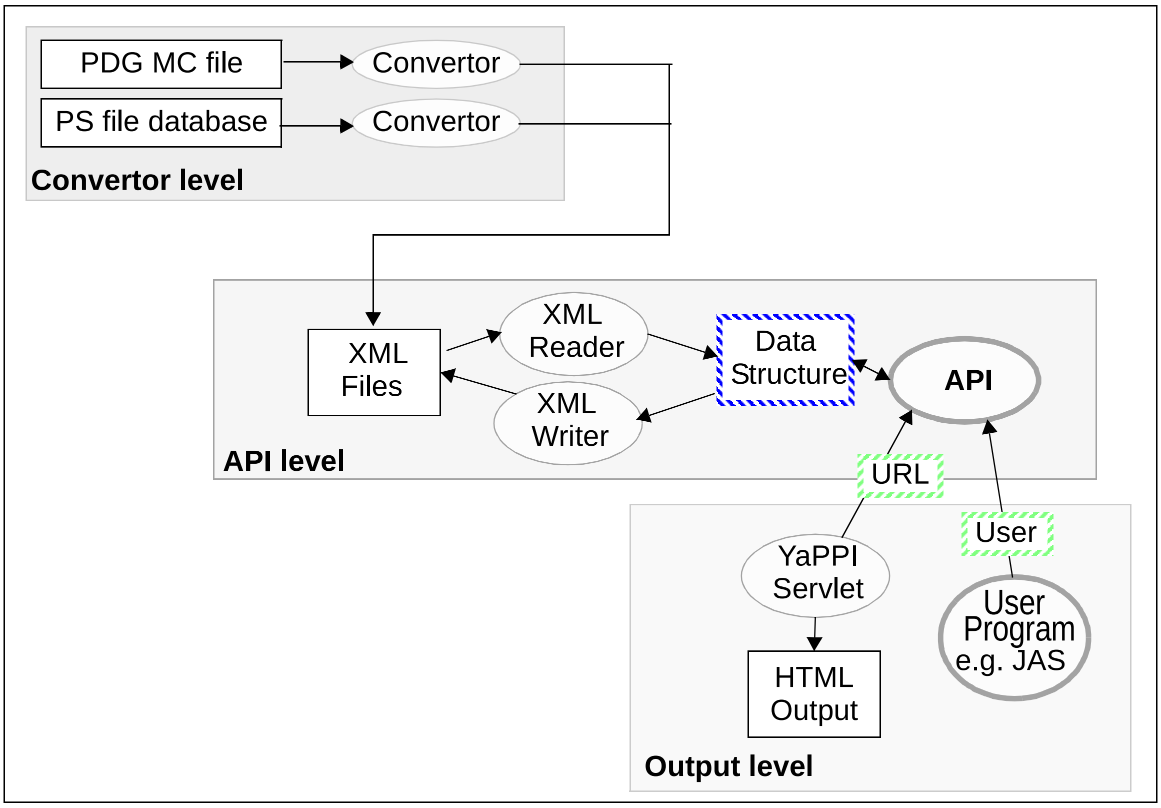


Figure 2: YaPPI data flow.

4. Implementation

Implementation consisted mostly of defining a class structure and an XML schema describing the data model. We built a program using the Apache Xerces Java Parser to read the XML files, using the event-based parsing model[7] to build a data structure in memory. Programs can use various methods to access particle data in memory.

For YaPPI's XML schema (see Figure 1), we have defined two main elements, or tags:

1. Family
2. ParticleType

The particle family tree is stored in a recursive way. Each family tag has as many sub-family entries and references to particle names as needed. The API looks the particles up in the XML file and makes a link to the proper particle. After reading the whole XML file a tree of the whole family can be accessed. This allows to browse through the particles in a very instructive way. An example of a family tree is shown in Figure 3.

The <ParticleType> defines a particle (see Figure 4). It has attributes describing the name of the particle. For typesetting reasons we added a LaTeX2e encoded attribute, "texName". The data of a particle is stored with <Data> tags. Each data entry (e.g. mass, width, etc.)

7. Simple API for XML (SAX).

```
<Family name=“Particles">
    <Family name=“Leptons">
        <ParticleType name=“e"/>
        <ParticleType name=“nu"/>
    </Family>
    <Family name=“Hadrons">
        <Family name=“Baryons">
            <ParticleType name=“N"/>
            <ParticleType name=“Delta"/>
        </Family>
        <Family name=“Mesons">
            <Family name=“Bottom">
            </Family>
        </Family>
    </Family>
</Family>
```

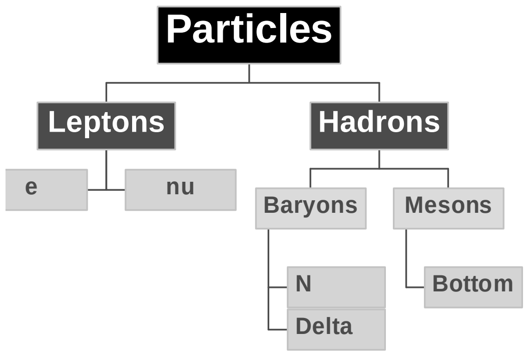


Figure 3: Example family tree (on left is the XML, and on the right is the resulting tree structure).

has one tag. A data tag consists of name, value, errors, unit, and more specific information.

The decay channels are stored separately with <Decay> tags. Each decay channel gets one decay tag. A decay channel consists of specific values like fraction, confidence level and errors and the products resulting out of the decay. A particle decays into a set of particles. In some cases only a subset of the decay is known, such as the family or possibly even less information. You can find some examples in Figure 5.

```
<ParticleType name="pi+”
      "texname="\pi^{\pm}">
   <Data  name=”Mass”
          value=”139.57”
          unit=”MeV”        />
   <Data  name=”Meanlife”
          value=”2.6e-8”
          unit=”s”          />
   </ParticleType>
```

Figure 4: A Particle entry in the XML file

The possibility to read multiple XML files allows for user defined data. You could first read a common PDG data file, then overwrite some of the data with your own experimental data.

All Particles known:	$D^{\pm} \longrightarrow \mu^{+} \nu\mu$
Particle+unknown:	$D^{\pm} \longrightarrow e^{+}$ anything
Family:	$W^{+} \longrightarrow$ hadrons

Figure 5: Examples for different decay channels

5. Data Access

Currently there are several sources of information:

1. A computer readable file from the PDG [PDG-MC] containing only a subset of information:
   a. Particle ID Number
   b. Particle Name
   c. Mass + Errors

    d. Width + Errors
    e. Charge
2. PDG Review [RPP] book in printed form
3. Database to create the PDG Review [RPP]
4. Postscript files from the PDG server containing the printed book in electronic form
5. XML files created by any source

We implemented a Java Application that read the computer readable file and created an XML file containing this limited set of information. We worked on a PostScript import utility and the results are convincing: Until now the PostScript import utility can get the LaTeX name, the masses, width, etc. The work will be continued to read also the decay channels.

A future task could be an administration utility, where you can search, import, export, etc. through the whole XML database.

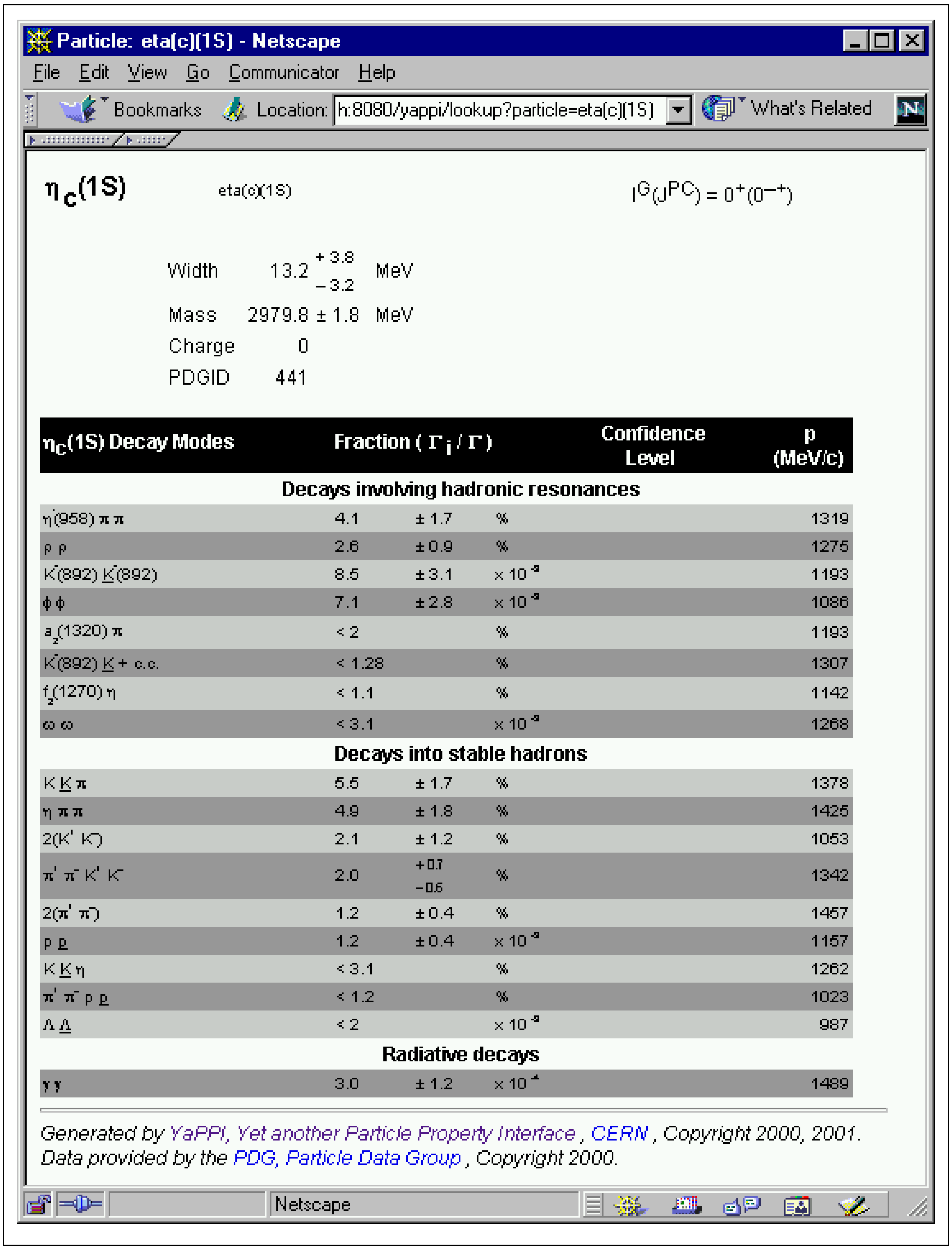
6. Conclusions

YaPPI provides particle properties to a wide range of applications. An API for Java applications it provides a language binding. Programs in other languages have to read the XML file itself, but there exist XML Parsers for nearly every programming language, including Fortran. YaPPI is platform independent and portable to many systems. The Servlet (see Figure 6) gives users an easy to use the particle property lookup program. The import utilities provide an interface to the PDG and therefore the guarantee that one can use the newest data.

Another advantage is the XML technology itself. Extending the data model will require only adding elements to the XML schema. This gives the possibility to add for example typesetting properties for printing the data.

References

JAS	Java Analysis Studio (JAS), http://jas.freehep.org
PDG	Particle Data Group, http://pdg.lbl.gov/
PDG-MC	Particle Data Group, computer readable file, http://pdg.lbl.gov/rpp/mcdata/garren_98.mc
RPP	*Review of Particle Physics*, Eur. Phys. J. C 15, 1-878 (2000)
Servlet	http://yappi.freehep.org
XERCES	Apache Xerces XML Parser for Java, http://xml.apache.org/xerces-j/index.html
XML	Extensible Markup Language (XML) 1.0 (Second Edition), W3C Working Draft 14 August 2000, http://www.w3.org/TR/2000/WD-xml-2e-20000814
XML98	Extensible Markup Language (XML) 1.0, W3C Recommendation 10-February-1998 http://www.w3.org/TR/1998/REC-xml-19980210


Particle: eta[c](1S) - Netscape
File Edit View Go Communicator Help
Bookmarks Location: h:8080/yappi/lookup?particle=eta[c](1S)
What's Related
Netscape

$\eta_c(1S)$ eta(c)(1S) $I^G(J^{PC}) = 0^+(0^{-+})$

Width	$13.2^{+3.8}_{-3.2}$	MeV
Mass	2979.8 ± 1.8	MeV
Charge	0	
PDGID	441	

$\eta_c(1S)$ Decay Modes	Fraction (Γ_i / Γ)			Confidence Level	p (MeV/c)
Decays involving hadronic resonances					
$\eta'(958)\pi\pi$	4.1	± 1.7	%		1319
$\rho\rho$	2.6	± 0.9	%		1275
$K^*(892)\underline{K}^*(892)$	8.5	± 3.1	$\times 10^{-3}$		1193
$\phi\phi$	7.1	± 2.8	$\times 10^{-3}$		1086
$a_2(1320)\pi$	< 2		%		1193
$K^*(892)\underline{K}$ + c.c.	< 1.28		%		1307
$f_2(1270)\eta$	< 1.1		%		1142
$\omega\omega$	< 3.1		$\times 10^{-3}$		1268
Decays into stable hadrons					
$K\underline{K}\pi$	5.5	± 1.7	%		1378
$\eta\pi\pi$	4.9	± 1.8	%		1425
$2(K^+K^-)$	2.1	± 1.2	%		1053
$\pi^+\pi^-K^+K^-$	2.0	$^{+0.7}_{-0.6}$	%		1342
$2(\pi^+\pi^-)$	1.2	± 0.4	%		1457
$p\underline{p}$	1.2	± 0.4	$\times 10^{-3}$		1157
$K\underline{K}\eta$	< 3.1		%		1262
$\pi^+\pi^-p\underline{p}$	< 1.2		%		1023
$\Lambda\underline{\Lambda}$	< 2		$\times 10^{-3}$		987
Radiative decays					
$\gamma\gamma$	3.0	± 1.2	$\times 10^{-4}$		1489

Generated by YaPPI, Yet another Particle Property Interface , CERN , Copyright 2000, 2001.
Data provided by the PDG, Particle Data Group , Copyright 2000.

Figure 6: The Java Servlet provides online information about the particle properties. Quick particle searches can be performed by specifying particle name or ID number. Particles may also be accessed by browsing through the family tree.