The Petri Net Markup Language (PNML) is an XML based interchange format for all kinds of Petri nets, which was published as International Standard ISO/IEC 15909-2:2011 in February 2011. Technically, ISO/IEC 15909-2 is defining an interchange format for three different kinds of high-level Petri nets and a simple version of Place/Transition systems only. But, one of the objectives of PNML was to provide a means for exchanging any kind of Petri net [8, 15, 1]. To this end, the concept of a Petri Net Type Denition (PNTD) was introduced, which is subject of a newly issued standardisation project ISO/IEC 15909-3. There are many tools supporting one form of PNML or the other, and, in particular, there is the PNML Framework [6], which helps tool developers to ease the implementation of PNML by providing a framework and an API for loading and saving Petri net documents in PNML. This framework is based on the Eclipse Modeling Framework (EMF) [2] and has the focus on the underlying meta-models of Petri nets. The PNML Framework, however, is not generic in the following sense: Whenever a new Petri net type is created, the code for the complete tool needs to be regenerated. Moreover, the PNML Framework does not come with a graphical editor for Petri nets. The ePNK overcomes these limitations: It provides an extension point, so that new Petri net types can be plugged in to the existing tool without touching the code of the ePNK. For defining a new Petri net type, the developer, basically, needs to give a class diagram (actually an ecore-diagram) dening the concepts of the new Petri net type, along with a mapping of these concepts to XML syntax. This type can then be plugged into the ePNK, and the graphical editor of the ePNK will be able to edit nets of this new type with all its features. Actually, this was the idea when we started the development of the Petri Net Kernel (PNK) about 15 years ago [12, 10, 13]. At that time, however, we had to implement all of the IDE functionality of such a tool ourselves. Today, we can make use of the eclipse platform [14], which helps us focusing on the Petri net specific parts; we get all the other functionality of a nice IDE, basically, for free. This is why we named the tool ePNK: it can be considered to be an eclipse based Petri Net Kernel. But, it is just the spirit and idea that the PNK and the ePNK have in common; technically, there is not a single line of code from the PNK in the ePNK, and they are not compatible. What is more, we use the nice features of EMF, GMF, and Xtext for developing the ePNK in a model-based way. In this way, the complete development process of the ePNK, is a case study in model-based software engineering using EMF and related technologies. This, actually, was the driving force behind this project. The evaluation and the lessons learned during this project, however, will be reported at an other occasion and to a different audience. This manual will focus on how to use the ePNK as an end user, and it will show how a developer can use the extension mechanisms of the ePNK for providing new Petri net types along with their XML syntax, and how to add new functionality to the ePNK.