

**Level:** Advanced  
**Works with:** Discovery Server  
**Updated:** 01-May-2003

by Wendi Pohs  
and [Dick McCarrick](#)

One of the more obvious and visible components of Discovery Server is the K-map feature. The K-map presents a taxonomy of all your content spidered and processed by Discovery Server. By using the K-map, you can see and navigate through your collected corporate documents and other forms of knowledge. This is especially useful if the K-map is organized in a way that appears logical and natural to users.

However, the K-map that works well for one group may not be equally well-suited for another, especially if that group is already familiar with an existing way of organizing and finding information. This is why virtually all Discovery Server customers modify the initial K-map produced by the system before rolling it out to their users. This can take a significant amount of editing, especially if you already have a clear idea of how you want your K-map to look. There are ways to help minimize the editing you need to perform on the K-map. For example, you can carefully tweak the settings of the K-map Building Service, the part of Discovery Server that actually builds the K-map. You can use these settings to help "guide" the K-map Building Service to produce a K-map closer to what you have in mind. For even more control, you can create a file system taxonomy and import it into Discovery Server. The system then uses this taxonomy as the blueprint for the K-map. A third option is to create a custom classifier. You can then use your classifier to build the K-map, instead of the K-map Building Service that comes with the product. This classifier can follow explicit rules you set, ensuring that the initial K-map it builds is already well on the way to the taxonomy scheme you want and that new documents will land in the categories you expect.

This article tells you how to do this. We start with a brief overview of ways to structure a taxonomy, particularly rules-based schemes. We then discuss classifiers and how they work, followed by two examples of how a site may approach organizing rules-based taxonomies. The remainder of the article describes how to create your own rules-based classifier. We look at the eClassifier tool and at sample code and explain what each does. You can use either method to build a classifier suited for your own site. (You can download the complete sample code in the [Discovery Server 2.0.1 API Toolkit](#).)

This article assumes that you're an experienced Java programmer familiar with Discovery Server and taxonomy/classification concepts in general.

## Taxonomy schemes

An important component of any full-featured knowledge management system is the taxonomy. This forms a graphical representation of your collective corporate documents, data, expertise, and other forms of knowledge. Well constructed taxonomies help you find the right people and information you need to answer a question or research a project. In Discovery Server, taxonomies are called K-maps. The K-map is a critical part of Discovery Server's search-and-browse user interface. The K-map helps you find content from many different sources by searching the subject categories. The K-map also displays information about relationships among documents, people, and categories, helping you browse and search for information in context. And after your taxonomy has been defined, Discovery Server uses it to determine affinities among users, document authors, and categories.

There may well be an infinite number of possible ways to organize a given collection of information into a

taxonomy. Some taxonomy schemes may work better than others depending on certain factors—the type of content, how the taxonomy is used, what its users are familiar with, and so on. In all likelihood, there's no one right way to structure a taxonomy. The scheme created for one audience may be completely inappropriate for another.

In approaching this problem, taxonomists have developed two general taxonomic structures called natural and rules-based. Natural schemes attempt to categorize data the way users logically use it. Rules-based taxonomies follow a specific set of predefined rules. In general, natural taxonomies are more free-form and informal, while rules-based ones are more structured. For example, imagine a Web site dedicated to woodworking. A natural way to organize content may be to group together documents describing how to set up your workshop, which tools to buy, what wood to buy, and how to build a few simple projects. A rules-based taxonomy might instead present all content related to tools in one category, different species of wood in another, and so on.

A lot of taxonomists may not agree with this assessment, but in our experience, natural taxonomies seem to work really well for some users, but not for others. This may be because of the different ways people think about and approach a problem. If the natural scheme follows the way you work, you'll find it very effective. If it doesn't, you may think it poorly organized and illogical. Rules-based taxonomies, on the other hand, tend to work at least reasonably well for most users. The structure may not follow the working and thinking process you use, but if well constructed, the rules are usually easy to understand and describe. And most of us (perhaps without even knowing it) have at least a little familiarity with rules-based taxonomies—for example, the Dewey Decimal classification system used by most public libraries.

## Classifiers

In the days before computers, taxonomies were built by people familiar enough with the subject matter being organized to spot relationships and affinities among the different documents. They had policies and procedures that told them where new documents should be filed. Nowadays, computers can do this automatically, using a program called a classifier. For example, Discovery Server builds its taxonomies using clustering software called the K-map Building Service (also known as the K-map Builder). In general, the clustering software builds a taxonomy by placing similar documents into categories, labeling these categories, and then placing related documents into the categories. When you click Discovery Server's Categorize button, its classifier comes into play. Its classifier analyzes the categories the clustering software has created and tries to add new documents to the categories where they fit best.

After your K-map Building Service builds the taxonomy, you can edit it to make it more user-friendly. This is an important step because even the most sophisticated knowledge management software can't join ideas and concepts the way humans can. It can also be one of the most time-consuming in the taxonomy roll-out process as you and your content experts rearrange categories and documents into a structure that feels logical and natural for your users. The K-map taxonomy produced by the K-map Building Service classifier may need a great deal of modification before it assumes the shape you want. This may not necessarily be the fault of the classifier, however. You may have some very specific ideas of what you want your K-map taxonomy to look like—ideas the classifier probably can't anticipate in advance. For example, you may want a strict rules-based taxonomy scheme or one that follows an existing file/folder format with which your users are already familiar.

Obviously, it would save you time and effort if the initial K-map taxonomy produced by Discovery Server already resembled the structure you have in mind. One relatively simple way to do this is to create a file system taxonomy and to import it into Discovery Server. This lets you "hard-wire" the categories into the scheme you want and lets Discovery Server fill it in with the appropriate content. This can be a useful option for some sites and offers your users the advantage of working with a hierarchy they already know. For more information on how to do this, see the *LDD Today* article "[Importing a file system taxonomy into a K-map](#)."

Alternatively, you can build your own rules-based classifier for Discovery Server. You can then build your K-map taxonomy with this custom classifier instead of the K-map Building Service. This can provide you with more control over how your K-map is constructed, following rules you specify to organize your categories and content. This way the initial K-map produced by Discovery Server can be much closer to the final working K-map you envision, requiring significantly less editing and modification before being deployed to your users. And new documents will be placed into categories according to the rules you describe.

## Two examples of rules-based classification

Before we consider options for rules-based classification, let's look at two sample user scenarios. As any good taxonomist knows, it's wise to do a little content analysis before building a taxonomy. Because you're usually working with many different repository types, you want to take advantage of any existing structure in the documents in these repositories, usually in the form of fields or document properties. If your content already contains many descriptive fields and is well organized, you may want to have Discovery Server maintain this

organization when it builds and classifies documents in your K-map. Or you may want to build your categories based on certain keywords that you know exist within the document body. You may be familiar enough with your subject area to determine these keywords in advance. Following are a couple of examples.

### **Scenario 1: Creating categories by using keywords in text**

A major consumer electronics manufacturer wants to use Discovery Server to provide precise search capability for its internal research papers, which are located on the file system, and for several major consumer packaging-related journals, which are located on the Web. The IT department also wants to identify common concepts for their researchers. The IT department has identified a content expert in each area. This person knows how he has organized content in the past, and he knows the terms he uses to search the external journals. He is a scientist, and he doesn't want to waste his time wading through reams of research reports. But he also doesn't want to have to depend on somebody else's organization of his data.

When asked, he can provide specific examples in his area of how he would group documents together. This expert tends not to rely on meta-data because the researchers all share the same Word document template. Instead, he likes to organize categories based on the occurrence of words, phrases, and synonyms in text. For example, he wants to create a specific category based on whether or not the phrases Palm and PDA occur together more than twice in the same document. If so, he wants the software to create a Palm Pilot category. If the term handheld device occurs more than once in a document, on the other hand, he wants the software to place that document into a Handheld Devices category. He would be comfortable using a tool like the K-map Editor to organize his hierarchy which would look like this:

Handheld Devices  
Palm Pilot

He also has documents that discuss multiple technologies. For these, he wants the category to be based on occurrences of multiple technology terms in the same document. For this category, the rule would look for occurrences of words like wireless and synchronization and Bluetooth. If they're all present in a document, he wants the software to place that document in a category called Research Plans. This category would then be on the same level of the hierarchy as the Handheld Devices category.

The IT department agrees with this methodology and expects to find content experts in other areas who can define rules the way this one content expert has. They envision each content expert defining a series of rules that they want to put together to create their rules-based classifier and taxonomy.

### **Scenario 2: Creating categories based on database fields**

A major engineering company is building a special Notes database containing approximately 4,000 documents that they plan to use as the foundation of their Discovery Server taxonomy. They describe the database structure this way: Each document will be classified using three hierarchical topic fields. At the top level is Strategic Report Type, and underneath are three sub-categories: Engineering, Marketing, and Research. Under each of those three subcategories are five or six topics, and under each of those an average of five or six subtopics. Inside each of those subtopics will be about 50 engineering, marketing, and research study documents that contain Title, Date, Author, and Summary fields.

This company wants Discovery Server to automatically use the Engineering, Marketing, and Research fields, along with each associated set of subcategories, as the basis for its taxonomy. New documents should be classified according to the way they appear in the Notes database.

## **Options for rules-based classification**

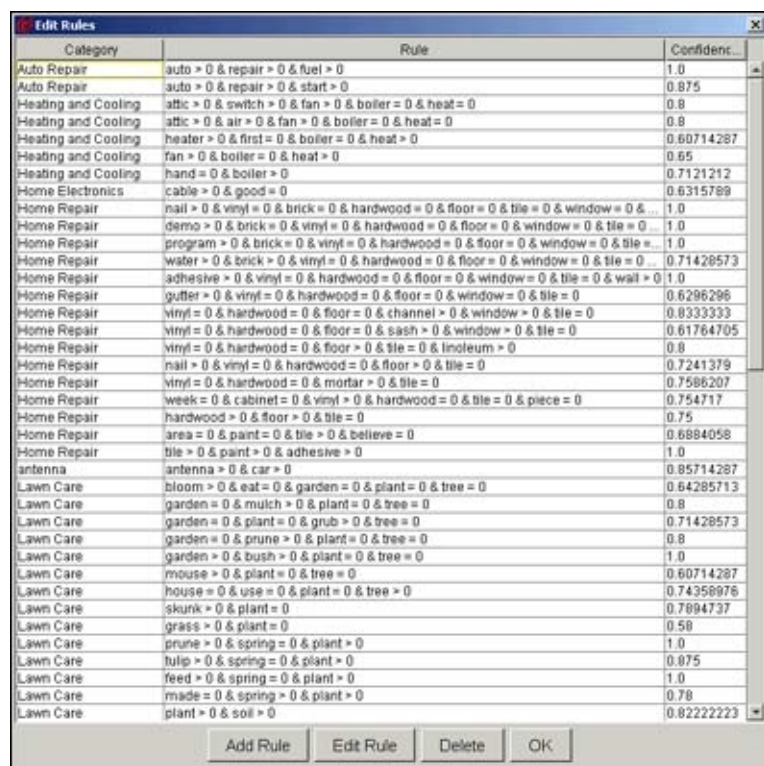
Neither of these scenarios would be particularly well served by the Discovery Server's K-map Builder and classifier because they rely on set rules that don't match the way Discovery Server builds taxonomies and classifies documents by default. But there are a couple of tools and sample programs that developers and administrators can use to create their own classifiers. We provide an overview of two of these, eClassifier and the KDSAPI Simple Classifier sample, in the sections that follow.

### **eClassifier**

eClassifier is a taxonomy analysis and editing tool that was developed at IBM Research. It includes many useful analytical and visualization features that taxonomists can use to create and refine taxonomies, as well as options for building many types of classifiers. We're going to concentrate on its rules-based capabilities in this article, but you can find complete information about eClassifier (including the eClassifier documentation) by visiting the [IBM Almaden Research Center eClassifier page](#).

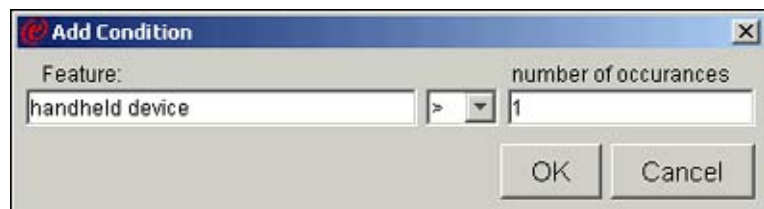
After the eClassifier software is installed locally, Discovery Server taxonomists can use it instead of or as a complement to the Discovery Server's K-map Editor. eClassifier's user interface provides a number of ways to create, review, modify, and edit the categories in a K-map.

If you have already generated a taxonomy using the Discovery Server's K-map Builder, eClassifier attempts to generate a series of rules for you. You download your existing Discovery Server taxonomy into the eClassifier and choose Edit - Edit rules. The rules for your K-map may look something like this:



Category	Rule	Confidence
Auto Repair	auto > 0 & repair > 0 & fuel > 0	1.0
Auto Repair	auto > 0 & repair > 0 & start > 0	0.875
Heating and Cooling	attic > 0 & switch > 0 & fan > 0 & boiler = 0 & heat = 0	0.8
Heating and Cooling	attic > 0 & air > 0 & fan > 0 & boiler = 0 & heat = 0	0.8
Heating and Cooling	heater > 0 & first = 0 & boiler = 0 & heat > 0	0.60714287
Heating and Cooling	fan > 0 & boiler = 0 & heat > 0	0.65
Heating and Cooling	hand = 0 & boiler > 0	0.7121212
Home Electronics	cable > 0 & good = 0	0.6315789
Home Repair	nail > 0 & vinyl = 0 & brick = 0 & hardwood = 0 & floor = 0 & tile = 0 & window = 0 & ...	1.0
Home Repair	dorm > 0 & brick = 0 & vinyl = 0 & hardwood = 0 & floor = 0 & window = 0 & tile = 0	1.0
Home Repair	program > 0 & brick = 0 & vinyl = 0 & hardwood = 0 & floor = 0 & window = 0 & tile = ...	1.0
Home Repair	water > 0 & brick > 0 & vinyl = 0 & hardwood = 0 & floor = 0 & window = 0 & tile = 0	0.71428573
Home Repair	adhesive > 0 & vinyl = 0 & hardwood = 0 & floor = 0 & window = 0 & tile = 0 & wall > 0	1.0
Home Repair	gutter > 0 & vinyl = 0 & hardwood = 0 & floor = 0 & window = 0 & tile = 0	0.6296296
Home Repair	vinyl = 0 & hardwood = 0 & floor = 0 & channel > 0 & window > 0 & tile = 0	0.8333333
Home Repair	vinyl = 0 & hardwood = 0 & floor = 0 & sash > 0 & window > 0 & tile = 0	0.61764705
Home Repair	vinyl = 0 & hardwood = 0 & floor > 0 & tile = 0 & linoleum > 0	0.8
Home Repair	nail > 0 & vinyl = 0 & hardwood = 0 & floor > 0 & tile = 0	0.7241379
Home Repair	vinyl = 0 & hardwood = 0 & mortar > 0 & tile = 0	0.7586207
Home Repair	week = 0 & cabinet = 0 & vinyl > 0 & hardwood = 0 & tile = 0 & piece = 0	0.754717
Home Repair	hardwood > 0 & floor > 0 & tile = 0	0.75
Home Repair	area = 0 & paint = 0 & tile > 0 & believe = 0	0.6884058
Home Repair	tile > 0 & paint > 0 & adhesive > 0	1.0
antenna	antenna > 0 & car > 0	0.85714287
Lawn Care	bloom > 0 & eat = 0 & garden = 0 & plant = 0 & tree = 0	0.64285713
Lawn Care	garden = 0 & mulch > 0 & plant = 0 & tree = 0	0.8
Lawn Care	garden = 0 & plant = 0 & grub > 0 & tree = 0	0.71428573
Lawn Care	garden = 0 & prune > 0 & plant = 0 & tree = 0	0.8
Lawn Care	garden > 0 & bush > 0 & plant = 0 & tree = 0	1.0
Lawn Care	mouse > 0 & plant = 0 & tree = 0	0.60714287
Lawn Care	house = 0 & use = 0 & plant = 0 & tree > 0	0.74358976
Lawn Care	skunk > 0 & plant = 0	0.7894737
Lawn Care	grass > 0 & plant = 0	0.58
Lawn Care	prune > 0 & spring = 0 & plant > 0	1.0
Lawn Care	tulip > 0 & spring = 0 & plant > 0	0.875
Lawn Care	feed > 0 & spring = 0 & plant > 0	1.0
Lawn Care	made = 0 & spring > 0 & plant > 0	0.78
Lawn Care	plant > 0 & soil > 0	0.82222223

Notice that each rule consists of a word or phrase, and the number of times that word or phrase should exist in the text of the documents in order for that document to appear in that category. You may agree with the rules that eClassifier has chosen, or you may decide, like the scientist in our first scenario, that there are additional rules that you want to create. In our scenario, the scientist knows he wants to create categories based on the occurrences of specific words or phrases in text. Using eClassifier, he highlights the category he wants and adds his rule or condition:

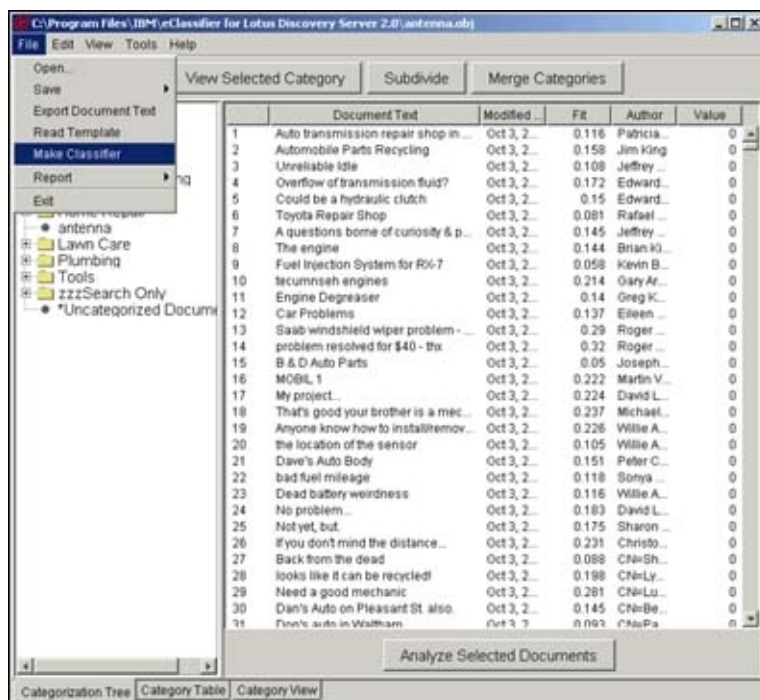


**Add Condition**

Feature:  number of occurrences:

OK Cancel

After he has finished, he can tell eClassifier to add documents to this new category, based on his new rule. He can spot-check the documents that eClassifier adds to this category, so he's sure the category doesn't include anything unexpected. And he can use eClassifier's other visualization tools to see if there are any other categories already in his taxonomy that are similar to the new one he has just created. He may want to revise his work if there are. After he has done this for all the categories he needs, he can tell eClassifier to build a new classifier for Discovery Server to add new documents to.



He can also save his work locally, so he can test his rules before he uploads or commits them to the server. After he uploads his new classifier to the Discovery Server and an administrator activates it, new documents appear in categories according to the rules he created.

### The Simple Classifier sample program

The Discovery Server 2.0.1 API Toolkit contains a sample program that experienced Java developers can modify to create a rules-based classifier. While eClassifier provides its own editing and analysis UI, the Simple Classifier sample program allows developers to hard-code the rules they know their users need. This could be a useful approach for the users in our second scenario. Because the taxonomy is based on known fields in a Notes database, developers can create a classifier that hard-codes the user instructions about these fields into rules. Unlike eClassifier, this approach provides no user interface for editing and modifying categories after the fact. Because the rules for building categories are well-known and accepted, developers can use this new classifier in conjunction with Discovery Server's K-map Building Service, and no additional K-map editing is needed.

But the Simple Classifier sample relies heavily on the existing Discovery Server services. It creates its own queue where the Discovery Server spiders, or data collectors, can place information about new documents that need to be classified. Because the Discovery Server uses XML to represent documents internally, the Simple Classifier program also depends on this representation of documents. The complete Simple Classifier sample with its detailed methods for creating work queues is available in the [Discovery Server 2.0.1 API Toolkit](#). We'll concentrate on its rule-creation section here.

After it has set up its internal queues, the Simple Classifier program describes its classification rules in its ApplyRules method. This is the section developers can modify to add their own content-specific rules. In this case, the developer has imported the standard W3 Document Object Model (DOM), so he knows which elements to expect. After he finds the elements and items he needs, he categorizes documents by company name, as long as the author has included the terms IBM, Lotus, Microsoft, or Oracle in the body of the document. Like the users in our second scenario, this developer already knows which repository his documents will come from. And he assumes that the categories he needs already exist in his K-map.

Here's where he parses the XML representation of his documents and looks for the words IBM, Lotus, Microsoft, or Oracle in the body field. The developers in our second scenario could modify this sample to look for fields containing Education, Marketing, or Research instead.

```
if (elementName.equals("FIELD"))
```



```
{
    org.w3c.dom.Node attrNode = attrList.getNamedItem("NAME");
    String attrValue = null;
    if (attrNode != null)
    {
        attrValue = attrNode.getNodeValue();
        if (attrValue.equalsIgnoreCase("FORM"))
        {
            form = childNode.getFirstChild().getNodeValue();
            if (form != null && form.length() == 0)
                form = null;
        }
        else if (attrValue.equalsIgnoreCase("SUBJECT"))
        {
            subject = childNode.getFirstChild().getNodeValue();
            if (subject != null && subject.length() == 0)
                subject = null;
        }
        else if (attrValue.equalsIgnoreCase("BODY"))
        {
            temp = childNode.getFirstChild().getNodeValue();
            if (temp != null)
            {
                if (temp.indexOf("IBM") != -1
                    || temp.indexOf("Lotus") != -1
                    || temp.indexOf("Microsoft") != -1
                    || temp.indexOf("Oracle") != -1)
                    body = temp;
            }
        }
    }
}
```

Rules-based classifiers can be as simple or as complex as the needs of the users of the taxonomy. After careful content analysis, you can decide if you want to have taxonomists define rules that developers can code for them or if you want them to create and modify their own categories, using sophisticated taxonomy editing tools like eClassifier.

## Conclusion

With your custom rules-based classifier in place, you have greater control over how your K-map is created and maintained. This in turn cuts down on the amount of editing you need to do before your K-map is ready to be deployed to your user community. By streamlining the K-map creation and roll-out process, you can expedite the adoption and use of Discovery Server throughout your organization—and more quickly start using this valuable tool to help make sense of all your corporate data.

## ABOUT THE AUTHOR

Wendi Pohs is a principal taxonomy specialist on the Discovery Server team and the author of a book about knowledge management methodologies, *Practical Knowledge Management: The Lotus Knowledge Discovery System*, published by IBM Press. Wendi joined Lotus Development Corporation in 1996 and has worked on various projects as a spec writer, online help designer, and user assistance manager. Prior to joining Lotus, Wendi worked at the American Mathematical Society and at Digital Equipment Corporation. Wendi received her BA and MILS degrees from the University of Michigan.