Endeca® Information Access Platform

Concepts Guide

December 2011

Contents

Copyright and disclaimer

Preface

Endeca® InFront enables businesses to deliver targeted experiences for any customer, every time, in any channel. Utilizing all underlying product data and content, businesses are able to influence customer behavior regardless of where or how customers choose to engage — online, in-store, or on-the-go. And with integrated analytics and agile business-user tools, InFront solutions help businesses adapt to changing market needs, influence customer behavior across channels, and dynamically manage a relevant and targeted experience for every customer, every time.

InFront Workbench with Experience Manager provides a single, flexible platform to create, deliver, and manage content-rich, multichannel customer experiences. Experience Manager allows non-technical users to control how, where, when, and what type of content is presented in response to any search, category selection, or facet refinement.

At the core of InFront is the Endeca MDEX Engine,™ a hybrid search-analytical database specifically designed for high-performance exploration and discovery. InFront Integrator provides a set of extensible mechanisms to bring both structured data and unstructured content into the MDEX Engine from a variety of source systems. InFront Assembler dynamically assembles content from any resource and seamlessly combines it with results from the MDEX Engine.

These components — along with additional modules for SEO, Social, and Mobile channel support — make up the core of Endeca InFront, a customer experience management platform focused on delivering the most relevant, targeted, and optimized experience for every customer, at every step, across all customer touch points.

About this guide

This guide introduces you to the Endeca Information Access Platform (Endeca IAP) and walks you through the Endeca development process.

After reading this guide, you will understand the key concepts underlying Endeca applications and be able to start building your own Endeca applications.

Who should use this guide

This guide is intended for developers who are new to the Endeca IAP, as well as individuals who want to understand the core concepts underlying an Endeca application.

Conventions used in this guide

This guide uses the following typographical conventions:

Code examples, inline references to code elements, file names, and user input are set in `monospace` font. In the case of long lines of code, or when inline monospace text occurs at the end of a line, the following symbol is used to show that the content continues on to the next line: ¬

When copying and pasting such examples, ensure that any occurrences of the symbol and the corresponding line break are deleted and any remaining space is closed up.

Contacting Endeca Customer Support

The Endeca Support Center provides registered users with important information regarding Endeca software, implementation questions, product and solution help, training and professional services consultation as well as overall news and updates from Endeca.

You can contact Endeca Standard Customer Support through the Support section of the Endeca Developer Network (EDeN) at *http://eden.endeca.com*.

Chapter 1

Welcome to the Endeca Information Access Platform

This section introduces the Endeca Information Access Platform.

Introduction

Endeca was founded on the simple idea that users want to explore data interactively in real time, relying on a friendly, intuitive interface, regardless of the scale and complexity of the underlying data.

Users need to search, navigate, and analyze all of their data, often in large and multiple data sources, "slicing and dicing" across any dimension, and drilling down to the finest grain of detail or zooming out to an aggregate view. At the same time, users need an application that responds intelligently to their current navigation state, guiding them along valid paths and eliminating invalid choices, or "dead ends." Users should experience simple, intuitive navigation even as they perform the equivalent of extremely complex, multi-dimensional database intersection queries.

The Endeca Information Access Platform (IAP), based on the Endeca MDEX Engine, is a powerful technology designed to help you build such easy and intuitive Guided Navigation applications. Guided Navigation not only tells users the results of their query, it also tells them all the valid "next-step questions" they can ask to refine and explore further, while eliminating the frustrating reply of "No Results Found". These next-steps are re-ranked and re-organized with each click, creating a much more productive and satisfying navigation experience for your users.

Endeca’s market solutions

Endeca offers a series of market solutions built on top of the Endeca Information Access Platform.

These solutions, such as Endeca for B2B eCommerce, Endeca for Web Site Search, and Endeca for Product Data Navigation, are packaged to meet specific information access business needs.

As each solution incorporates a different set of modules (for example, the Endeca for Retail Analytics solution includes visualization and analytics functionality), not all of the features described in the documentation may be available as part of your Endeca license. However, because these solutions are based on the same platform, much of the functionality is identical across solutions.

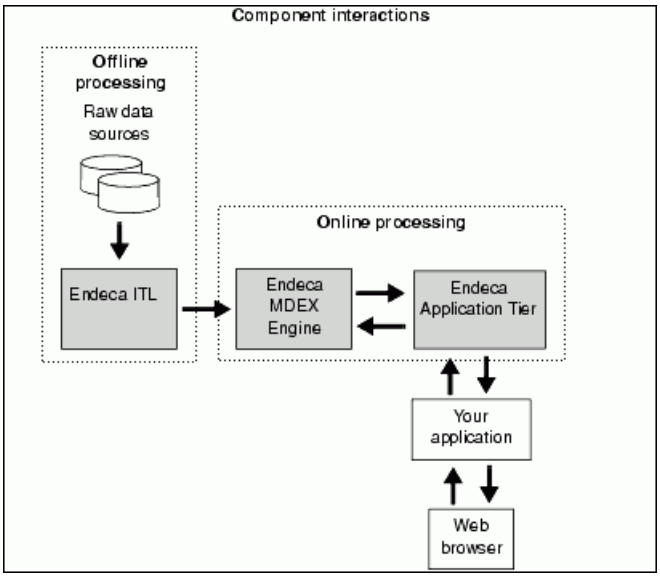
Endeca IAP components

The Endeca IAP has three major components.

These components are:

- Endeca Information Transformation Layer (ITL)
- Endeca MDEX Engine
- Endeca Application Tier

These components interact with your data sources and application as shown in the following figure.



The Endeca Information Transformation Layer (ITL) reads your raw source data and manipulates it into a set of Endeca MDEX Engine indices. The ITL consists of the Content Acquisition System (which includes the Endeca CAS Server and Console, the CAS API and the Endeca Web Crawler), and the Data Foundry (which includes data-manipulation programs such as Forge).

The Endeca MDEX Engine is the query engine that is the core of the Endeca IAP. The MDEX Engine consists of the Indexer (Dgidx), the Dgraph, and the Agraph. The MDEX Engine loads the indices generated by the indexing component of the Endeca Information Transformation Layer. Although the Indexer (also known as Dgidx) is installed as part of the MDEX Engine package, in effect it is part of the ITL process.

After the indices are loaded, the MDEX Engine receives queries from the Endeca Application Tier, executes them against the loaded indices, and returns the results to the client Web browser. The Application Tier provides an interface to the MDEX Engine. The two default interfaces, which can be used in the same application, are the Presentation API and the Web services interface. The Endeca RAD (Rapid Application Development) Toolkit for ASP.NET can also be used. These interfaces are used to query the MDEX Engine and manipulate the results.

The Endeca ITL components, such as Forge, are run offline at intervals that are appropriate for your business requirements. The Endeca MDEX Engine and Endeca Application Tier are both online processes, meaning they must remain running as long as you want clients to have access to your data set.

Installation packages for Endeca components

The Endeca components are installed in several packages.

This section provides general information about the installation packages. For detailed information on all installation packages for the Endeca IAP, and their related documentation, see the *Endeca Getting Started Guide*.

Platform Services

The Platform Services core installation package includes components such as Forge, the Endeca Application Controller (EAC), and the Endeca Logging and Reporting System. For more detailed information, see the *Platform Services Installation Guide*.

MDEX Engine

The MDEX Engine core installation package installs the Endeca MDEX Engine. The server on which the MDEX Engine will run requires the EAC Agent installed from the Platform Services installation package. For detailed information on installation requirements, see the *MDEX Engine Installation Guide*.

Endeca Workbench

The Endeca Workbench is a suite of tools that brings together Web-site management capabilities and provides features for system administrators to configure the resources used by an Endeca implementation, monitor its status, start and stop system processes, and download an implementation's instance configuration for debugging and troubleshooting purposes.

For detailed information on installation requirements, see the *Endeca Workbench Installation Guide*.

Application Tier

The Application Tier consists of parts that belong to different installation packages. The Application Tier is the component that sends queries to the MDEX Engine and processes the query results. The two Endeca default interfaces that interact with the MDEX Engine are:

- Presentation API (available as its own installation package on the MDEX Engine download page). For information on the Presentation API, see the *Endeca Basic Development Guide* and the *Endeca Advanced Development Guide*.
- Web services and XQuery for Endeca (available as part of the MDEX Engine package). For details, see the *Endeca Web Services and XQuery Developer's Guide*.

You can also use the Endeca RAD Toolkit for ASP.NET package.

Endeca MDEX Engine query results

All query results returned from the Endeca MDEX Engine contain two types of information.

These information types are:

- The appropriate results for the query (for example, a record set or an individual record)
- The supporting information for building follow-on queries

The follow-on query information allows users to refine or broaden their query and, correspondingly, their query results. The method the MDEX Engine uses to compute this information eliminates invalid follow-on queries (*dead ends*). Eliminating dead ends and providing relevant next-step refinement

choices are two of the primary features that distinguish Endeca solutions from other types of search implementations.

Two types of queries

The Endeca IAP provides two types of queries: navigation queries and keyword search queries.

- *Navigation queries* return a set of records based on application-defined record characteristics (such as wine type or region in an online wine store), plus any follow-on query information.
- *Keyword search queries* return a set of records based on a user-defined keyword, plus any follow-on query information. (There is a second type of keyword search query as well, described in Chapter 4.)

Navigation queries and keyword search queries are complementary. In fact, a keyword search query is a specialized form of navigation query, and the data structures for the results of the two queries are identical: a set of records and follow-on query information.

Users can execute a combination of navigation queries and keyword search queries to navigate to their desired record set in the way that works best for them. For example, users can execute a keyword search query to retrieve a set of records, then use a follow-on navigation query to refine that set of records. The reverse situation is also valid.

Logical versus physical structure

The data in an Endeca application has both a physical structure and a logical structure that support your MDEX Engine queries.

To understand the difference between physical structure and logical structure, consider a relational database. A relational database has a set of tables, each of which contains its own data. Relationships exist among the tables that allow you to create logical records from data spread across multiple tables. The database's physical structure is a set of individual tables, and its logical structure is a set of records whose data is drawn from those tables.

An Endeca implementation also imposes both a physical structure and a logical structure on your data. The next chapter describes these structures and how the MDEX Engine uses them to respond to queries.

Reference implementations

In addition to the three basic Endeca IAP components, your Endeca distribution also contains a set of reference implementations that implement many of Endeca's features.

You used the basic wine reference implementation when you tested your Endeca installation.

You can use these optional reference implementations as a starting point or guide when building your own application. The reference implementations incorporate two types of information:

- *Data reference implementations* show you how to use the Information Transformation Layer features to convert your source data into MDEX Engine indices.

- *UI reference implementations* show you how to add Endeca features to your application's user interface.

The reference implementations are located in the reference directory of your Endeca Platform Services installation. For more information about the reference implementations, see the *Endeca Basic Development Guide*.

Chapter 2

Understanding Records, Dimensions, and Properties

This section explains Endeca properties and dimensions, as well as the structure of Endeca records.

About Endeca records, dimensions, and properties

In order to understand Endeca applications, you must understand three basic constructs: Endeca records, dimensions, and properties.

Endeca records and dimensions provide the logical structure for the data in your Endeca application. This structure supports both navigation and search queries. Endeca properties provide descriptive information about individual Endeca records. This section introduces these three constructs, and describes their relationships to each other.

Endeca records

Endeca records are the entities in your data set that you are navigating to or searching for.

Bottles of wine in a wine store, customer records in a CRM application, and mutual funds in a fund evaluator are all examples of data stored as Endeca records.

Endeca records generally correspond to traditional records in a source database. Unlike source records, however, Endeca records have been standardized for consistency and classified with dimension values, which are described below. This classification is a key step toward building the logical structure for your Endeca application.

An Endeca record may correspond to multiple records in your source data. For example, you could have four source records that refer to the same book in a variety of formats: hardcover, paperback, large print, and audio. You can build an Endeca application so that these four individual records correspond to a single Endeca record.

Dimensions and dimension values

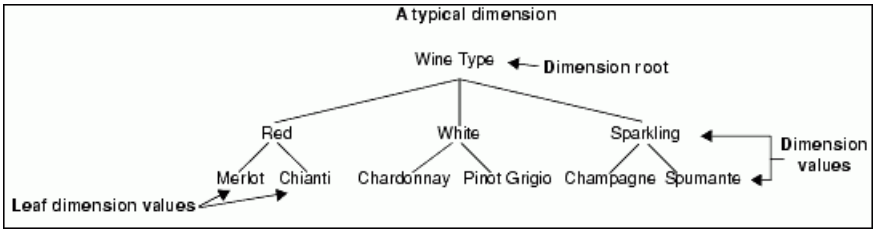
Dimensions provide the logical structure for organizing the records in your data set.

Your Endeca application can have many dimensions, and dimensions can be hierarchical.

A dimension is a collection of related dimension values, organized into a tree. The top-most dimension value in a dimension tree is known as the *dimension root*. A dimension root always has the same name as its dimension.

The bottom-most dimension values in the tree are referred to as leaf dimension values. You can think of dimension values as "locations" within a dimension tree.

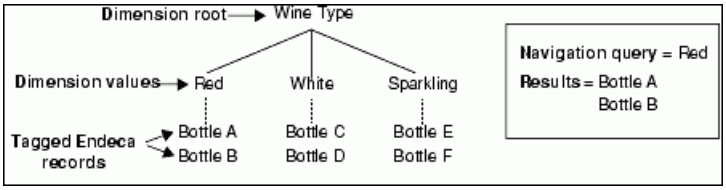
Note: The term "location" in this dimension value definition is used strictly in a logical sense. The records in an Endeca application are not physically structured into trees.



Dimension values are tags, or labels, you use to classify the records in your data set. Tagging a record with a dimension value does the following:

- It organizes the record within the tree structure of the associated dimension. In the example below, Bottles A and B are organized under the Red dimension value in the Wine Type dimension, while Bottles C and D are organized under the White dimension value, and so forth.
- It identifies the record as a valid result when that dimension value is selected in a navigation query.

In the example below, a navigation query on the Red dimension value produces a result set of Bottles A and B.



In essence, these two statements are equivalent. You can think of a navigation query as "return all the records that are organized under the xxx dimension value in the yyy dimension." This is the logical view. Or, you can think of it as "return all the records that have been tagged with the xxx dimension value in the yyy dimension." This is the physical view. In either case, the results are the same.

A record can be tagged with many dimension values from many dimensions. This allows users to navigate to records through any dimension or combination of dimensions they choose. (Using multiple dimensions is described in more detail in the "Multiple dimensions" topic.

Related Links

Multiple dimensions on page 19
Your Endeca application can have as many dimensions as needed, and each record in your data set can be tagged with zero, one, or more dimension values within each dimension.

Mapping source properties

Endeca records are derived from source records.

Generally, a source record is nothing more than a set of key/value pairs, or source properties, that contain descriptive information about the record. For example:

Record example

Record ID: 0001
Name: House White
Wine Type: Chardonnay
Vineyard: Sonoma Vineyards
Year: 1996
Price: $45.00
Rating: 96
Description: Intense, with complex earthy pear, fig, melon, citrus and hazelnut flavors that are remarkably elegant and sophisticated.

A record's source properties provide the basis for how the record will be displayed, searched for, and navigated to in an Endeca application.

Transforming source records to Endeca records

During the transformation of a source record into an Endeca record, one of three things happens to each of the source record's properties.

The three property transformations are:

- You map the source property to a dimension.

  Mapping a source property to a dimension effectively tags the record with a dimension value from that dimension. As described in the section above, dimensions and dimension values are the backbone of Endeca's navigation functionality.

- You map the source property to an Endeca property.

  Endeca properties are intended for display once the end user has searched for or navigated to a record set or an individual record. Mapping a source property to an Endeca property enables the source property for use within your Endeca implementation. It is important to understand that, while Endeca properties can be displayed and searched, they *cannot be used for classification and navigation*. You can think of Endeca properties as descriptive information only.

- You decide to ignore the source property (for using it in navigation).

  If a source property does not contain information for navigation, search, or display, you can instruct your Endeca implementation to ignore it during the data transformation process.

After all the properties of a source record have been mapped, the source record has been transformed into an Endeca record consisting of Endeca properties and dimension value tags.

Comparing Endeca properties and dimensions

While Endeca properties and dimensions share many similarities, properties do not support navigation.

Endeca properties and dimensions are similar in that they both:

- Are usually generated from a record's source properties, using source property mapping.
- Consist of key/value pairs (property name/property value, dimension name/dimension value).
- Can be searched and displayed.

The primary difference between Endeca properties and dimensions is that the MDEX Engine indices for dimensions support navigation, while those for Endeca properties do not.

When deciding how to map a source property, consider whether you want to build a navigation query based on that property. For example, a Wine Type source property is useful for navigation purposes

because it allows you to identify a set of wines according to type: red, white, sparkling, and so forth. This type of source property should be mapped to a dimension.

A Description property, on the other hand, whose value contains a lengthy description of the wine, would not be appropriate for a navigation query. You wouldn't expect an end user to query for "all wines that are intense, with complex earthy pear, fig, melon, citrus and hazelnut flavors that are remarkably elegant and sophisticated." It is much more appropriate to display this type of information after the end user has navigated to the record. This type of source property should be mapped to an Endeca property.

Another difference between the two is that Endeca properties often contain more specific information about a record than dimensions. For example, a Price Range dimension is useful for navigation—give me all the bottles of wine that cost between $10 and $20 dollars—but it's the exact price of each bottle that you want to see when looking at the individual records. A common implementation for this type of application uses a Price Range *dimension* for navigation and a Price *property* that is displayed once a bottle's record has been located.

Note: Property, dimension, and dimension value names are case sensitive. If you intend to use the Endeca Query Language, property and dimension names must conform to the NCName convention. See "NCName format for properties and dimensions" in the *Endeca Advanced Development Guide* for more information on this restriction. In addition, the XQuery for Endeca feature also requires that dimension names follow the NCName format when used with the dimension-related functions.

Automatically deriving dimensions from source data

Dimensions are often, although not always, derived automatically from source data properties.

The following two-dimensional table gives an example of what your source data might look like:

Source data

Source properties

	Wine Type	Country	Price
Bottle A	Red	USA	$15
Bottle B	Red	Chile	$25
Bottle C	White	France	$18
Bottle D	White	Chile	$54
Bottle E	Sparkling	USA	$35
Bottle F	Sparkling	France	$22

Source records

Property values

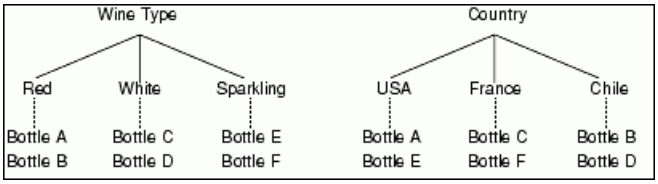
Note: While it is most convenient to use a tabular format for your source data, other formats are also acceptable. For details, see the *Endeca Forge Guide*.

In this source data table, the following relationships exist:

- Each row corresponds to a source record. Each source record can be transformed into an Endeca record.
- Each column represents a property. Each property can be mapped to a dimension, an Endeca property, or both.

- The column entries represent the property values that are assigned to each record within each source property. These values map to either dimension values (if the source property is mapped to a dimension), or Endeca property values (if the source property is mapped to an Endeca property).

With automatic dimension generation, source properties become dimensions and source property values become dimension values. If we mapped the Wine Type and Country source properties to dimensions, and then automatically generated those dimensions, the logical representation of the generated dimensions would look like this:



The Wine Type and Country source properties become the Wine Type and Country dimensions. Each source property value becomes a dimension value within its respective dimension: Red, White, and Sparkling for the Wine Type dimension and USA, France, and Chile for the Country dimension. The individual records are organized beneath their respective dimension values accordingly.

With automatically-generated dimensions, it is important to remember that changes to your source data's properties and property values directly influence those dimensions and their dimension values.

Note: The Endeca Information Transformation Layer allows you to build very flexible data pipelines for transforming your source data. For example, you may choose to map only some of your source properties to Endeca properties and dimensions, or you may add Endeca properties and dimensions that don't exist in your source data at all. The above discussion is intended only as a general guideline for the relationship between source data and automatically generated dimensions. Ultimately, you have total control over the entire transformation process from source data to MDEX Engine indices.

More about dimensions

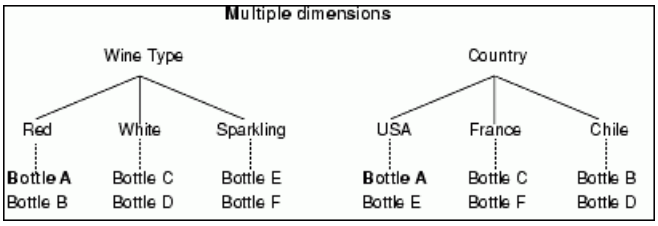
Now that you understand the basics of Endeca records, Endeca properties, and dimensions, it's time to explore dimensions a bit further.

This section describes additional dimension concepts that you should understand before building an Endeca application.

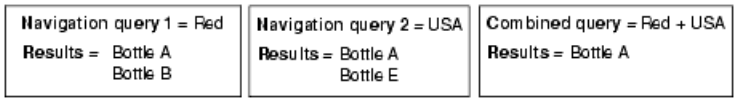
Multiple dimensions

Your Endeca application can have as many dimensions as needed, and each record in your data set can be tagged with zero, one, or more dimension values within each dimension.

In the illustration below, Bottle A is tagged with the Red dimension value in the Wine Type dimension, and with the USA dimension value in the Country dimension.



By using multiple dimensions to classify your Endeca records, you introduce the possibility for complex navigation queries across multiple dimensions, like those in the following example:



In this example, a *navigation query* was used. A navigation query is a query that returns a set of records based on user-selected characteristics along with any follow-on information.

Multiple dimensions also allow you to navigate to a record using any dimension tagged to the record, in any order. The navigation queries in both of the following examples produce the same results:

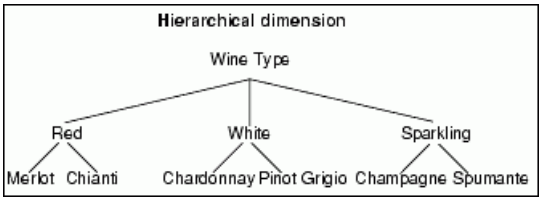
Dimension hierarchy

Dimension hierarchy gives you additional control over the logical structure used to organize your Endeca records.

As the term "dimension tree" implies, dimension values can have *parent* and *child* dimension values.

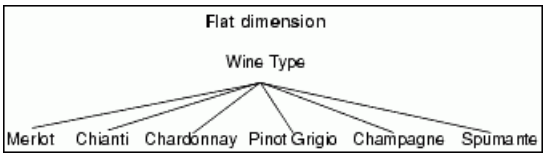
A dimension value that has sub-dimension values is the *parent* of those sub-dimension values. The sub-dimension values themselves are *children* or *child dimension values*. Child dimension values of the same parent dimension at the same level of hierarchy are *dimension value siblings*.

This is what a typical hierarchical dimension looks like:



Child dimension values are always more specialized than their parents. Child dimension values help users to further refine their navigation query and, consequently, the resulting record set (see the topic "Refining a navigation query in a hierarchical dimension" for a more detailed explanation of query refinement).

Dimensions that have only one level of hierarchy beneath the dimension root are called *flat* dimensions. The following illustration shows a flat version of the Wine Type dimension.



Dimensions that are automatically generated are always flat.

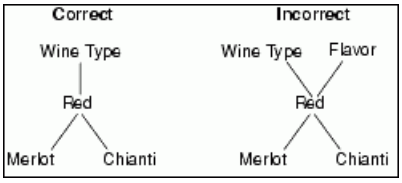
Related Links

Refining a navigation query in a hierarchical dimension on page 21
When you navigate a dimension value, you are also implicitly navigating all of its children.

The one parent rule

It is possible for a dimension value to be simultaneously a child of one dimension value and the parent of other dimension values.

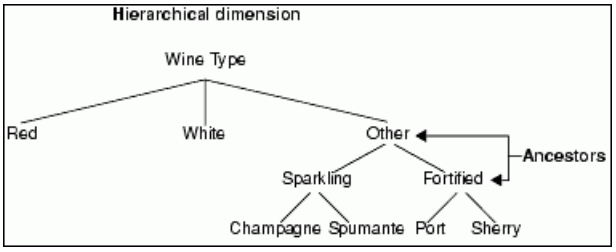
Each dimension value, however, can have only one parent. In the example on the left below, Red is the child of the Wine Type dimension value and the parent of the Merlot and Chianti dimension values. The example on the right is incorrect, because Red is specified as the child of both the Wine Type and Flavor dimension values. This organization is not allowed in an Endeca dimension hierarchy.



Ancestors

In addition to parents, dimension values can have ancestors.

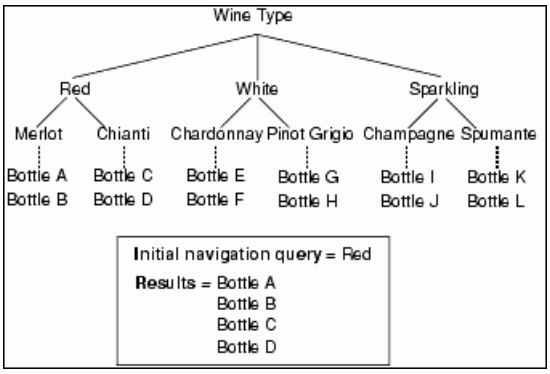
Ancestors represent the dimension values between the dimension root and your current location in the dimension tree (technically, a parent is an ancestor). In the example below, Other and Fortified represent the ancestors for the Sherry dimension value.



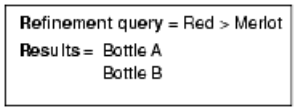
Refining a navigation query in a hierarchical dimension

When you navigate a dimension value, you are also implicitly navigating all of its children.

This means that the MDEX Engine returns not only the records tagged with the dimension value you specified, but also those records that are tagged with any of the dimension value's children. In the following example, a navigation query on Red would produce a result set of four bottles, Bottles A through D.



If we refine the query by navigating to one of Red's children (Merlot, for example), the result set is cut in half, in this case to Bottles A and B.



Navigating from a dimension value to one of its children refines the navigation query and typically reduces the size of the result set. Conversely, navigating from a dimension value to the value's parent, or ancestor, typically broadens the query and increases the size of the result set. It is the information required to create these refining and broadening queries that makes up the bulk of the follow-on query information contained in the MDEX Engine's query results.

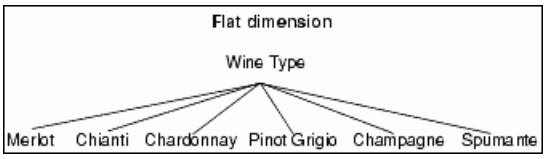
Note: Due to their refining nature, child dimension values are also often referred to as refinement values or refinements.

Because navigation queries can cross multiple dimensions, the MDEX Engine's query results contain follow-on information for every dimension. This means that you can refine or broaden your query in one or more dimensions while making no change to the query parameters in the other dimensions.

Advantages of dimension hierarchy

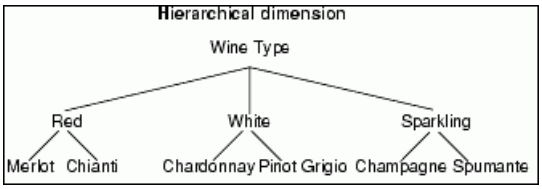
Dimension hierarchies allow you to exercise a higher level of control over the number of follow-on queries that are presented to users as they navigate.

For example, in the flat dimension below, a navigation query on the Wine Type dimension value would return six possible refinement queries, one for each child.



This is a simple example, but a large flat dimension with many dimension values is too unwieldy to navigate. The user would be presented with too many potential follow-on queries.

Using hierarchical dimensions reduces this information overload and provides for an easier, more intuitive navigation experience. In the hierarchical example below, the Wine Type dimension value has only three possible refinement queries: Red, White, or Sparkling.



A second reason to use dimension hierarchy is that, by limiting the number of refinement queries, you decrease the amount of time it takes for the MDEX Engine to return its results. Returning query results that contain follow-on information for 20 refinement queries is faster than returning query results that contain information for 2000 refinement queries.

Note: The decrease in performance is due to network transfer and page size issues, not MDEX Engine computational speed.

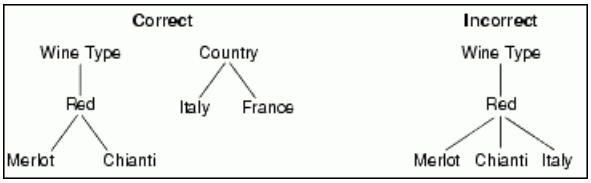
Additional dimension hierarchy recommendations

Creating intuitive dimension hierarchies is critical to providing a comfortable navigation experience for your end users.

his section provides some additional advice about what you can do to keep your hierarchies as intuitive as possible.

Using a consistent theme

All of the dimension values in a dimension should be part of the same theme. This is the case for all dimensions, but it becomes particularly important when you are building hierarchical dimensions.

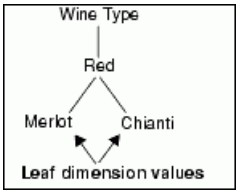


Because navigating on a dimension value implicitly navigates all of the dimension value's children, a child dimension value that does not have the same theme as its parent can result in a confusing set of Endeca records with multiple themes.

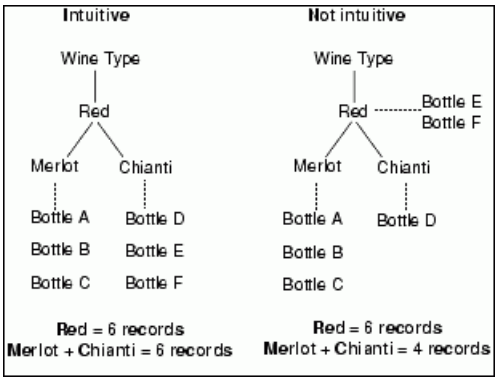
Also, child dimension values that are not more specialized than their parents can create nonsensical navigation paths for the user. In the incorrect example above, refining the Wine Type dimension value by navigating to the Red dimension value feels correct and intuitive. Refining by navigating to the Italy dimension value, however, does not because the two values are not closely related to each other.

Tagging Endeca records with leaf dimension values

While it is technically possible to tag your Endeca records with any dimension value that exists in a dimension, it is highly recommended that you tag your Endeca records with leaf dimension values.



Tagging your Endeca records with leaf dimension values creates a more intuitive navigation experience, because the sum of the records in the leaf dimension values will equal the number of records returned for the parent dimension value.



Creating dimensions and tagging records

You build dimensions and their hierarchies in Endeca Developer Studio.

The actual tagging of records with dimension values occurs when you process your source data into a set of MDEX Engine indices, using the Endeca Information Transformation Layer.

Alternatively, you can set up your Endeca application to derive dimensions automatically from the source data's properties. Remember, however, that automatically derived dimensions are always flat. Hierarchical dimensions must be created manually in Developer Studio.

Chapter 3

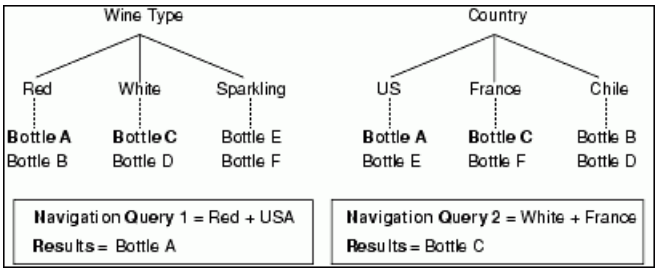
Understanding Guided Navigation

This section describes how Endeca's Guided Navigation provides a powerful and intuitive experience to users as they navigate through a data set.

The anatomy of a navigation query

In its most basic form, a navigation query is a combination of one or more dimension values.

These dimension values are referred to as the *navigation descriptors*. A navigation query instructs the Endeca MDEX Engine to return the set of records that represents the intersection of all the dimension values that it contains. For example, in the illustration below, Bottle A represents the intersection between the Red and USA dimension values. Bottle C represents the intersection between the White and France dimension values.



When an intersection does not exist between all of the dimension values in a navigation query, that query is considered a dead end. For example, in the illustration above, the Sparkling and Chile dimension values have no bottles in common and, therefore, no intersection. (In other words, there are no sparkling wines from Chile.)

The MDEX Engine automatically removes the possibility of such dead-end queries by the way it structures the follow-on query information that it returns in its query results. This is the essence of Guided Navigation.

Note: A default navigation query uses all of the dimension values together (that is, it implies an AND operation). This is the type of query we are describing here. The Endeca IAP has features that allow you to implement other, more advanced types of queries. See the "Default and advanced navigation queries" topic.

Related Links

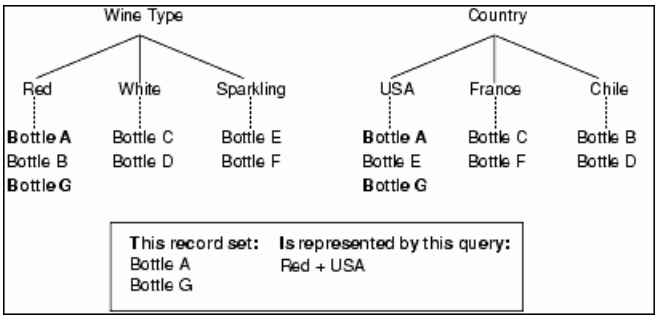
Default and advanced navigation queries on page 26

Default navigation queries allow only one dimension value per dimension.

Identifying a record set using a navigation query

To extend the concept introduced in this section one step further, a specific record set can be represented by a combination of dimension values.

The example below adds another bottle of wine to our data set, Bottle G. Bottle G is tagged with the same dimension values as Bottle A (Red and USA). Executing a query on Red + USA would produce a record set including both Bottles A and G. Therefore, the record set of Bottles A and G can be represented by the navigation query "Red + USA".



Default and advanced navigation queries

Default navigation queries allow only one dimension value per dimension.

The following are examples of default navigation queries:

```
Red
Red + France
Red + France + 1996
```

In order for a record to be returned for a default query, it must be tagged with all of the dimension values in the query. In other words, a default navigation query is an AND query.

The Endeca MDEX Engine also supports more advanced queries that contain multiple dimension values from a single dimension. These values can be used in AND queries or OR queries. For example:

```
(Oak AND Berry) + 1996
Red + (France OR Chile)
```

The Endeca MDEX Engine does not support the use of OR queries across dimensions. In other words, a query for *France OR Red* is not allowed.

See the *Endeca Basic Development Guide* for more information on Endeca's advanced multi-select query options.

About Guided Navigation

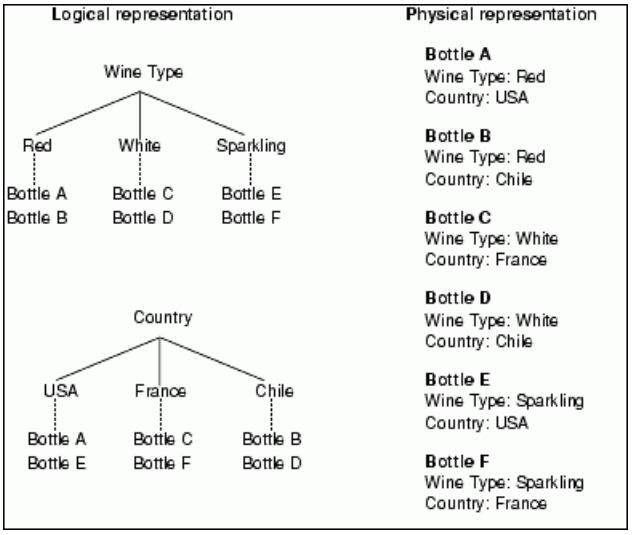
Guided Navigation is the presentation of valid follow-on refinement queries to the user.

You can think of Guided Navigation as the elimination of invalid refinement queries, or dead ends. To understand how the MDEX Engine accomplishes Guided Navigation, we'll look at the refinement information that the MDEX Engine returns in response to a query.

Note: Unless otherwise stated, everything said in this section pertains to both navigation queries and record search queries (a type of keyword search query described in the "Record search" topic).

The refinement information that is returned for a query is *specific to the record set that is returned for the query*. For example, if a returned record set contains only red wines, refinements associated with white or sparkling wines are not returned. It is the record data that is returned for a query, therefore, that drives which refinements are also returned with the query.

As explained in the "Dimensions and dimension values" topic, you can think of a dimension logically as a tree of dimension values. This structure is useful because it makes it easy to imagine a specific location within a dimension. It also makes it easier to see the intersections among dimension values. Physically, however, the tagging of records with dimension values happens at the record level. This means that the illustration on the right is a more accurate way of representing the data in the MDEX Engine than the trees we've seen until now.



Every query returns a set of records that are tagged with dimension values, much like the physical representation above. From this representation, the MDEX Engine can quickly determine which dimension values are tagged to the returned records and which are not. This dimension value tagging determines what follow-on query information is returned along with the set of records:

- Any dimension value that is tagged to at least one record in the returned record set is considered a valid refinement (because querying the dimension value would return at least one record). The follow-on refinement queries for these dimension values are returned in the query results.
- Dimension values that are not tagged to any records in the returned record set are considered invalid. No follow-on query information is returned for these dimension values.

  To understand this concept better, remember that querying a dimension value returns the set of records that have been tagged with that dimension value. If none of the records in a record set have been tagged with a particular dimension value, then querying that dimension value would produce no results. This makes the dimension value an invalid refinement option.

Tagging the data with dimension values and adjusting the follow-on refinement queries in this way allows the MDEX Engine to dynamically build navigation paths among the Endeca records in a data set. Users can navigate along any path supported by the current record set, and the precise set of relevant paths updates as the record set changes. This is the essence of Endeca's Guided Navigation.

Related Links

Record search on page 33
A record search query is Endeca's equivalent to full-text search.

Dimensions and dimension values on page 15
Dimensions provide the logical structure for organizing the records in your data set.

Guided Navigation example

This section illustrates the Guided Navigation concept in the context of a typical Endeca application.

In the example, we use a simplified version of the wine UI reference implementation you used to test your Endeca installation. It also illustrates the use of dimension hierarchy.

Note: The example in this section deals solely with navigation queries. See the "Using Keyword Search" section for more information on record search queries, and using navigation queries and record search queries together.

Related Links

Using Keyword Search on page 33
This section describes the two types of keyword search queries: record search and dimension search.

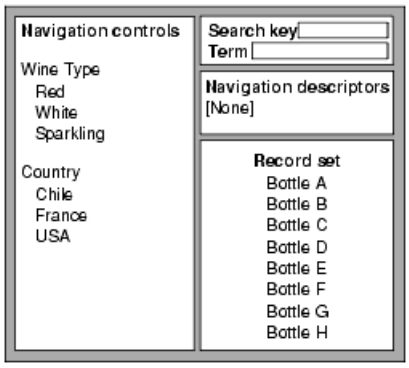
A typical Endeca application

The Endeca Presentation API supports a wide variety of application features and user interface styles.

Commonly, however, the primary page of an Endeca application has these four components:

- Some type of navigation controls that contain the dimension values the user can select to navigate around the data set.
- A record set that is made up of all the Endeca records that are valid results for the current query.
- A set of navigation descriptors that tells the user which dimension values, if any, were used in the query that returned the current record set.
- Some type of search controls that allow users to perform keyword searches.

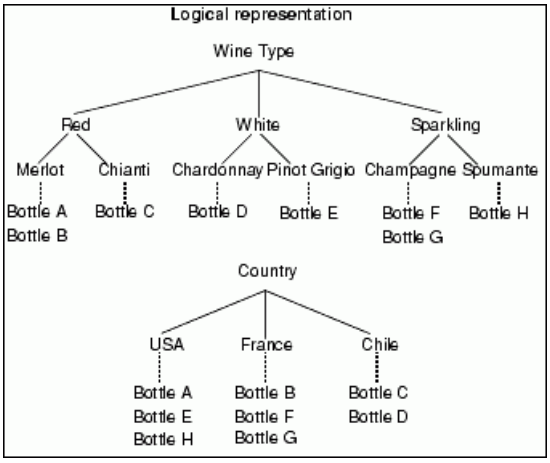
The following illustration shows a typical primary page for an Endeca application.



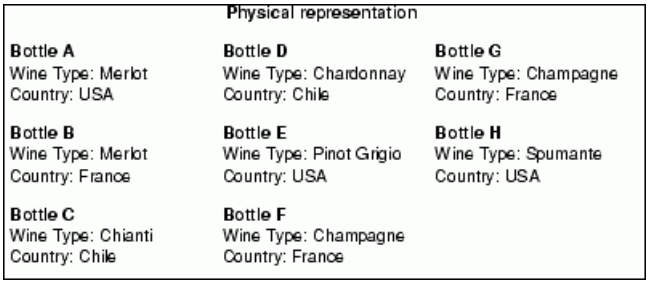
Source data

This topic shows the logical and physical representations of the wine store data.

The logical representation of the wine store data looks like the following:



The physical representation of the wine store data looks like this:

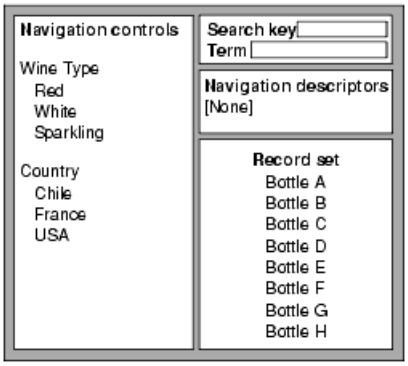


Working through the example

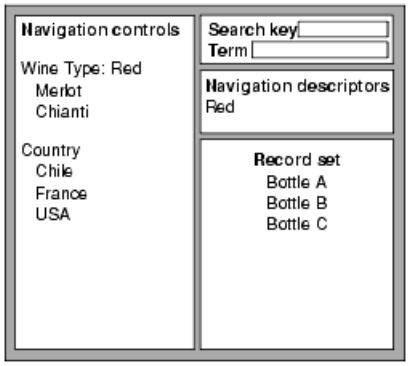
In all of the UI reference implementations, you start by looking at the entire data set (this is also known as starting at the root of the data set).

Starting at the root means that all of your Endeca records are displayed, and all of your refinement values are displayed, organized by dimension.

In the case of our wine store example, Bottles A through H are displayed in the Endeca record set, and the refinements in the Wine Type and Country dimensions are displayed in the navigation controls.



Step 1: We begin refining the list of bottles by selecting the Red dimension value from the Wine Type dimension. The Endeca Presentation API sends a navigation query containing the Red dimension value to the MDEX Engine which returns the query results.



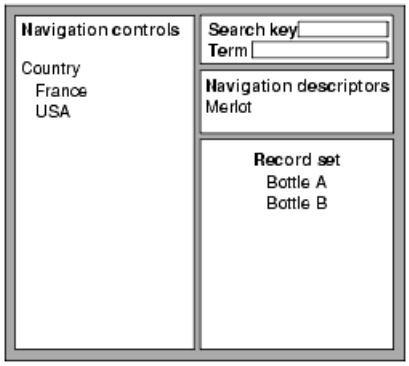
The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with the Red dimension value or any of its children.

  Bottles A, B, and C are tagged with children of the Red dimension value so they remain in the record set. Bottles D through H are excluded.

- Red is added to the navigation descriptors area, indicating that the navigation query used to retrieve the current record set contained the Red dimension value.
- The navigation controls change to reflect the current record set, according to the rules of Guided Navigation:
  - Bottles A and B are tagged with Merlot, and Bottle C is tagged with Chianti, so those dimension values remain in the navigation controls.
  - None of the bottles are tagged with White or Sparkling, so those dimension values are omitted from the Wine Type dimension hierarchy. (This makes sense because a bottle of wine cannot be red and white, or red and sparkling.)
  - The Country dimension remains unchanged because Bottles A, B, and C are tagged with USA, France, and Chile, respectively.

Step 2: Next, we can refine our navigation query by selecting the Merlot dimension value.



The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with the Merlot dimension value.

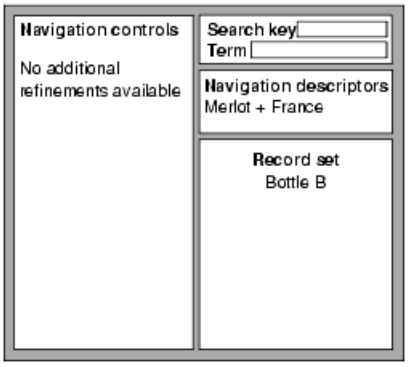
  Bottles A and B are tagged with Merlot, so they remain in the record set. Bottle C is excluded.

- The Red dimension value is refined to Merlot in the navigation descriptors area, indicating that the navigation query used to retrieve the current record set contained the Merlot dimension value.

  Note: Default queries only allow one dimension value per dimension, so Merlot replaces Red in the query.

- The navigation controls change to reflect the current record set:
  - The Wine Type dimension is omitted from the navigation controls because all of the records in the current record set are tagged with Merlot, and Merlot is a leaf dimension value that cannot be further refined. In other words, the Wine Type dimension no longer needs to be displayed because there are no refinement values left to pick from it.
  - The Country dimension changes to reflect the current record set:
    - Bottles A and B are tagged with USA and France, respectively, so those dimension values remain in the navigation controls.
    - Neither bottle is tagged with Chile, so the Chile dimension value is omitted from the Country dimension hierarchy.

Step 3: Next, we can select France from the Country dimension to see the effects of navigating across multiple dimensions.



The sample application displays the following behavior:

- The record set is refined to include only bottles tagged with both the Merlot and France dimension values.

  Bottles B is tagged with Merlot and France so it remains in the record set. Bottle A is tagged with Merlot and USA, so it is excluded.

- France is added to Merlot in the navigation descriptors area.
- The navigation controls change to reflect the current record set. In this case, the Country dimension is removed from the navigation controls because France is a leaf dimension value that cannot be further refined. This leaves our example with no more dimensions to navigate.

Chapter 4

Using Keyword Search

This section describes the two types of keyword search queries: record search and dimension search.

Record search

A record search query is Endeca's equivalent to full-text search.

Record searches return the following:

- A set of records based on a user-defined keyword(s).
- Follow-on query information, based on the returned record set.

Record search queries are performed against a particular property or dimension, also known as the *search key*. In order to perform a record search, the application's user:

1. Chooses an Endeca property or dimension to act as the search key.
2. Specifies a term, or terms, to search for within the key.

The set of records that a record search returns is made up of all records whose value for the search key contains the specified term(s). By default, if the record search query specifies multiple terms, a value must contain all of the terms in order for its record to be returned (the query is treated as an AND query).

Note: See the *Endeca Basic Development Guide* for information on advanced search queries using OR and Boolean logic.

Record search queries apply to a defined record set only (for example, the current record set). So a record search query is made up of two parts:

- A set of dimension values that identify the currently defined record set (in other words, a navigation query).
- The search key and term(s).

In essence, a record search query is a navigation query that is modified by a search key and terms. This is why the two queries, navigation and search, have identical data structures for their results: a set of records and follow-on query information.

Only properties or dimensions that have been configured for record search in Endeca Developer Studio can be used as search keys.

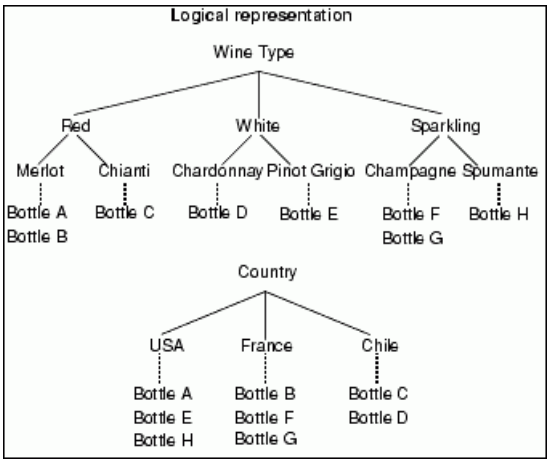
Using search interfaces

Endeca properties and dimensions that have been specified as searchable may also be combined into searchable groups called *search interfaces*. Search interfaces allow users to search across multiple properties/dimensions simultaneously.

Integrated navigation and record search example

This section illustrates integrating record search queries with navigation queries to navigate to a specific record set.

It uses the same sample application and source data from the "Guided Navigation example" topic. As a reminder, the logical representation of the wine store data from the Guided Navigation example looks like this:



The physical representation of the wine store data looks like this:

Physical representation

Bottle A Wine Type: Merlot Country: USA	**Bottle D** Wine Type: Chardonnay Country: Chile	**Bottle G** Wine Type: Champagne Country: France
Bottle B Wine Type: Merlot Country: France	**Bottle E** Wine Type: Pinot Grigio Country: USA	**Bottle H** Wine Type: Spumante Country: USA
Bottle C Wine Type: Chianti Country: Chile	**Bottle F** Wine Type: Champagne Country: France	

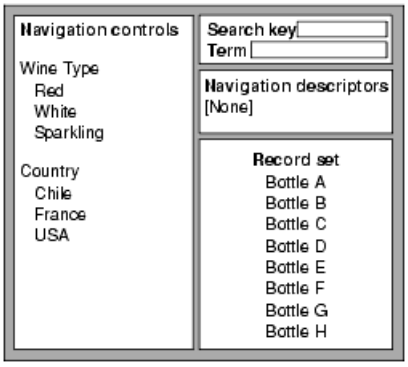
Related Links

Guided Navigation example on page 28
This section illustrates the Guided Navigation concept in the context of a typical Endeca application.

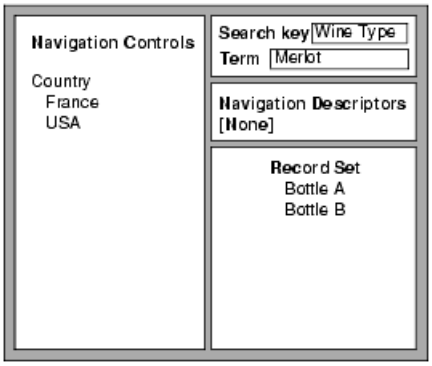
Working through the example

Again, in the UI reference implementations, we start by looking at the entire data set.

The figure below displays all of the Endeca records and all of the refinement values, organized by dimension.



Step 1: We begin by performing a record search query for *merlot* in the Wine Type dimension. A record search query containing the search key and search term (Wine Type and Merlot, respectively) is sent to the MDEX Engine and the query results are returned.



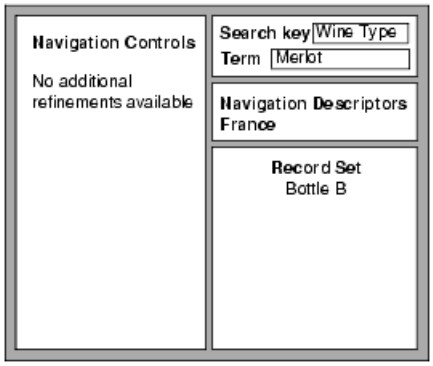
The sample application displays the following behavior:

- The record set is refined to include only bottles whose value for Wine Type includes (anywhere) the term *merlot*.

  Bottles A and B fit this description so they remain in the record set. Bottles C through H are excluded.

- The navigation controls change to reflect the current record set, according to the rules of Guided Navigation:
  - The Wine Type dimension is omitted from the navigation controls because all of the records in the current record set are tagged with Merlot, and Merlot is a leaf dimension value that cannot be further refined. In other words, the Wine Type dimension no longer needs to be displayed because there are no refinement values left to pick from it.
  - The Country dimension changes to reflect the current record set:
    - Bottles A and B are tagged with USA and France, respectively, so those dimension values remain in the navigation controls.
    - Neither bottle is tagged with Chile, so the Chile dimension value is omitted from the Country dimension hierarchy.

Step 2: Next, we can select France from the Country dimension to see the effects of navigating after performing a record search. A navigation query for France, that has been modified with the Merlot search term, is sent to the MDEX Engine.



The sample application displays the following behavior:

- The record set is refined to include only bottles that are:
  - Tagged with the France dimension value
  - Have a Wine Type value that includes the text *merlot*

  Bottle B fits this description so it remains in the record set. Bottle A is excluded.
- The navigation controls change to reflect the current record set. In this case, the Country dimension is removed from the navigation controls because France is a leaf dimension value that cannot be further refined. This leaves our example with no more dimensions to navigate.

Dimension search

In addition to record search, the Endeca MDEX Engine supports a second type of keyword search called dimension search.

Dimension search queries return dimension values that have names that contain search term(s) the end user has specified. Unlike record search, dimension search does not require a search key. Dimension search always searches any dimension values that have been identified as searchable for the terms provided. You identify a dimension as searchable in Endeca Developer Studio.

The dimension values that are returned for a dimension search are, by definition, navigable. This means that you can use the information in the dimension search results to build navigation queries, providing your users a seamless transition from a dimension search paradigm to a navigation paradigm.

Default and compound dimension searches

Dimension search has two modes: default and compound.

The difference between the two dimension search modes is:

- Default mode returns single dimension values, organized by dimension.
- Compound mode returns sets of dimension values.

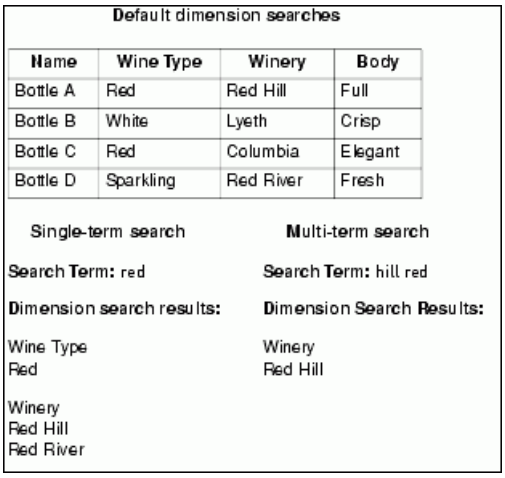
Default dimension search

Default dimension searches return individual dimension values, organized by dimension, for both single and multi-term queries.

If the user provides multiple terms, then a dimension value must contain *all* of the terms to be included in the results.

The example below illustrates both single term and multi-term default dimension searches. For the search term *red*, the MDEX Engine returns three dimension values: Red (from the Wine Type dimension), and Red Hill and Red River (from the Winery dimension). The dimension values are organized by dimension.

The multi-term default dimension search returns a single dimension value, Red Hill, because it is the only dimension value that contains both search terms.



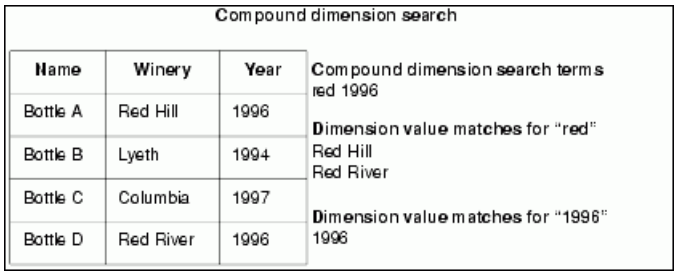
Compound dimension search

Compound dimension search extends the functionality of default dimension search by returning not single dimension values, but combinations of dimension values called navigation references.

A navigation reference is essentially a navigation query waiting to happen.

To start a compound dimension search, you provide the search terms. After the terms are provided, the following happens:

1. The MDEX Engine looks at each term individually, and searches for any dimension values that match the term. The result of this process is a list of dimension values.



2. The MDEX Engine combines the dimension values it found in Step 1 into navigation references, and tests the validity of those references.

   A navigation reference is valid if executing a query with it returns at least one record. In the example above, two navigation references are valid:

```
Red Hill + 1996
Red River + 1996
```

The MDEX Engine continues evaluating navigation references until it has assessed all possible combinations.

3. The MDEX Engine returns any valid navigation references it has found in Step 2.

Combining search queries

You can combine record search and dimension search queries into one consolidated MDEX Engine query.

Typically, you combine search queries to give your users a little extra information as they navigate through your site.

Combining search queries allows you to retrieve not only a refined record set with follow-on query information, but also a separate set of navigable dimension values based on the same keyword(s). You can present this separate set of dimension values to the end user as a more targeted list of potential follow-on queries (in addition to the standard follow-on queries that accompany a record search).

When you pass multiple types of search to the MDEX Engine in one query, the Endeca Presentation API treats each search as an individual request to the MDEX Engine. The search queries are completely independent of each other, and the MDEX Engine returns a result object for each search type. The Presentation API then combines these multiple result objects into one query result object.

Note: When combining record search and dimension search, keep in mind that record searches require a search key and one or more search terms. Dimension searches use only search terms.

Comparing dimension search and record search

Dimension search and record search each have their own strengths.

In general, you should:

- Use dimension search when the search terms are included in your dimension hierarchy.
- Use record search when you want to search unstructured data that is not part of the dimension hierarchy.

Dimension search is more appropriate than record search when the value for the search key does not consistently contain the search terms across all records. For example, not all red wines have "red" in their name, so a record search for *red* in the Name key would not return a complete set of red wine records. Dimension search, however, would return a Red dimension value that allows the user to navigate to a red wine record set (assuming you have tagged all red wines with a Red dimension value).

On the other hand, it is inappropriate to create dimension values out of certain types of property values that are long, wordy, or otherwise unsuitable (for example, descriptions or reviews). Because these property values are not included in the dimension hierarchy, they cannot be searched with dimension search. They must be searched using record search.

Additional search features

There are other search features that you can incorporate into your application.

This chapter gave you a brief overview of the types of search that you can implement in your Endeca application. There are many more search features that you can take advantage of, some of which are:

- Spelling functionality enables search queries to return expected results even though the user has misspelled the search term.
- Did You Mean functionality allows you to provide suggestions for further record searches to your users. Did You Mean is very useful when dealing with misspelled words or words that have exact matches that may not be as appropriate as other more popular alternatives.
- Stemming and thesaurus allow your Endeca application to consider alternate forms of individual words as equivalent for the purpose of search querying. For example, in many applications it is desirable for singular nouns to match their plural equivalents in the searchable text, and vice versa; this is an example of stemming. The Thesaurus feature allows the system to return matches for related concepts to words or phrases contained in user queries. For example, one might configure a thesaurus entry to allow searches for *Mark Twain* to match text containing the phrase *Samuel Clemens*.
- Relevance Ranking allows you to control the order in which results are returned. In particular, it is typically desirable to return results for the actual user query ahead of results for stemming and/or thesaurus transformed versions of the query.

See the *Endeca Basic Development Guide* for basic information on search and the *Endeca Advanced Development Guide* for implementing more advanced search features.

Chapter 5

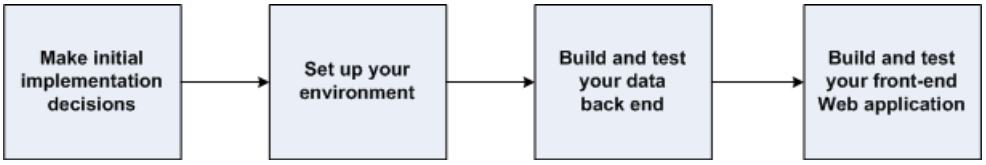
Understanding the Endeca Development Process

This section provides an overview of the major tasks you will perform while building an Endeca implementation.

Major development tasks

Developing an Endeca implementation can be broken down into four major tasks.

The four tasks are:



The following sections provide an overview of each major task.

Note: This chapter gives you a high-level view of the Endeca development process. Specific implementation details are covered by other areas of the documentation.

Making initial implementation decisions

There are two decisions you should make up front, before you begin any development on your Endeca implementation.

These two decisions are:

- Deciding where you will get your source data
- Choosing a Web application server

Getting your source data

The first step in any Endeca project is to locate your source data and answer a series of questions.

The questions you should answer are:

- Where does my raw source data come from? Do I have multiple data sources?

  If so, and the joins between the sources are simple, you can have the Endeca Data Foundry component do them. If the joins are complex, you should do them in your database management system before importing the data into the Data Foundry. A join combines records from two or more tables in a relational database.

- What data constitutes a record in my Endeca application?

  For example, an Endeca record could be a movie (encompassing all media formats) or it could be a movie in a particular format such as DVD, videotape, or laser disc.

- Which source properties do I need to maintain for display and search in my Endeca implementation? Which source properties should be excluded?
- What dimensions do I want to use for my navigation controls?

  In general, you should have a column (or field) in your source data for each dimension that you want to include in your Endeca implementation, although you can also add dimensions to your records as they are processed.

- What transformations must be done to standardize the properties and property values in your source data records for consistency?

  The Endeca Data Foundry functionality supports limited data cleansing. If you have an existing data cleansing infrastructure, it may be more advantageous to use those facilities instead.

After evaluating your source data, you can choose a mechanism for controlling your environment.

Choosing a Web application server

The Web application server you choose dictates which versions of the Presentation API you can use to communicate with the MDEX Engine.

Therefore, you should decide which application server you want to use before you begin implementing your front-end Web application.

Note: The simplest choice to make is to use the same Web application server you used during the basic Endeca installation because you know that server is already set up and functioning properly.

Setting up your environment

After making the initial implementation decisions, you can begin preparing your environment.

Environment preparation includes:

- Preparing your source data.
- Creating a supporting directory structure.
- Setting up your Web application server.

Note: A production environment may also include such things as load-balancing switches, redundant machines, firewalls, and so forth. These are advanced topics that are not covered in this chapter.

Preparing your source data

Your source data should be prepared as much as possible before it is processed by the Endeca ITL.

To prepare your source data, do the following:

- Perform any standardizing required in your database management system
- Implement any joins required in your database management system
- Output your source data in a format that is compatible with the Endeca Information Transformation Layer. The format of source data with which it is easiest to work is a two-dimensional, delimited format where each row corresponds to a record and each column corresponds to a source property. For details, see the *Endeca Forge Guide*.

Creating a directory structure

An Endeca implementation relies on a specific directory structure for such things as source data, working files, log files, and so forth.

See the *Endeca Forge Guide* for a description of typical directory structures.

Setting up your Web application server

Follow the vendor's instructions to set up and configure your Web application server.

If you are using the same Web application server that you used during the basic Endeca installation, then this step is already completed.

Building and testing your data back end

After making your initial implementation decisions and setting up your environment, you are ready to begin development.

Endeca recommends that you start with developing your data processing back end using the Endeca Information Transformation Layer (ITL). The result of ITL development is a running instance of a MDEX Engine, populated with indices built from your source data.

Once you have a running MDEX Engine, you can view the engine's data using one of the sample Web applications included with your Endeca distribution. These sample Web applications allow you to view, test, and debug your data processing back end before you have developed your front-end Web application. Decoupling the development of the data processing back end from that of the front-end Web application has two major benefits:

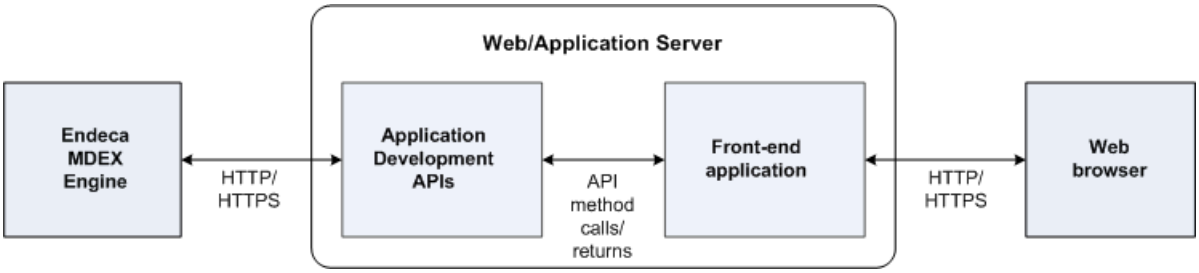
- It allows for early testing and debugging of the indices stored by the MDEX Engine.
- It increases development efficiency because you can spread development tasks among multiple developers.

The sample Web applications included with the Endeca distribution are called UI reference implementations.

Building and testing your front-end Web application

You develop a front-end Web application to an Endeca implementation in the same way you would any other Web application.

Connecting your Web application to the MDEX Engine is a simple matter of making calls to one of the Application Development APIs to request, retrieve, and manipulate the data contained in the Engine, as is illustrated in the following diagram:



The Application Development APIs could be one of the following:

- Presentation APIs
- RAD Toolkit for ASP.NET
- Content Assembler APIs
- XQuery

Note: The first three APIs in this list (Presentation APIs, RAD Toolkit for ASP.NET and Content Assembler APIs) include an associated UI reference implementation application.

The UI reference implementation will help you while you are coding your front-end Web application.

Performance testing and tuning

Performance testing and tuning are critical to developing an Endeca implementation that behaves well and provides your end-users with a satisfactory experience.

There are a number of factors that affect performance, including:

- The design of your Endeca implementation
- The infrastructure your implementation is running on, and
- The way you have used and configured the various Endeca features.

Endeca strongly recommends that you incorporate performance testing and tuning into your overall development schedule and has provided the Endeca *Performance Tuning Guide* to assist you with these tasks. You can find this guide in the knowledge base on the Endeca Developer Network (EDeN) site at: *http://eden.endeca.com*.

Index

M

N

O

P

Q

R

S

T

U

W