

---

# Electronic Catalog XML (eCX) Specification

## **XML for Catalog Interoperability**

---

Version 2.0

May 2000

## Overview

This document provides an outline of the eCatalog (eCX) Specification for electronic catalog interoperability and eContent interchange. This specification does not address transactions or other EDI definitions. It is based exclusively on the exchange of catalog information and multi-vendor catalog interoperability.

## Purpose

The purpose of the eCatalog XML specification is to provide a method of updating a catalog's structure, or schema, and its eContent from a variety of sources and content formats. The XML format has been designed to allow import of eContent into any catalog format. It supports the following functionality:

- Modification of catalog schema
  1. Add, update, and delete a category
  2. Add, update, and delete a common attribute, or a category attribute
- Modification of catalog products
  1. Add, update, and delete products
  2. Move products between categories
  3. Copy products to new categories
- Support for Internationalized data

## Specification

The design goal of this specification is to allow maximum flexibility by not specifying absolute tags, catalog schema, types, or formats in the XML document type definition (DTD).

### Document Type Definition (DTD)

```
<?xml version="1.0"?>
<!DOCTYPE CATALOG [

<!ELEMENT CATALOG ( ADMIN, (SCHEMA|DATA)* )+>
<!ATTLIST CATALOG xml:lang NMTOKEN #IMPLIED>

<!ELEMENT ADMIN (NAME, INFORMATION, NAMEVALUE*)>
<!ELEMENT SCHEMA (CATEGORY | ATTRIBUTE | NAVIGATION | METADATA)*>
<!ELEMENT DATA (ITEM | DATAEXTENSION | PRICING)*>

<!ELEMENT INFORMATION ( DATE, SOURCE, AUTHOR?, TITLE, TITLELOGO?, DOMAIN?)>
<!ELEMENT DATE (#PCDATA)>
<!ELEMENT SOURCE (#PCDATA)>
<!ELEMENT AUTHOR (#PCDATA)>
<!ELEMENT TITLE (#PCDATA)>
<!ELEMENT TITLELOGO (#PCDATA)>
<!ELEMENT DOMAIN (#PCDATA)>

<!-- Commonly used data elements -->
<!ELEMENT NAME (#PCDATA)>
<!ELEMENT KEY (#PCDATA)>
<!ELEMENT VALUE (#PCDATA)>
<!ELEMENT TYPE (#PCDATA)>
<!ELEMENT COMMENT (#PCDATA)>
<!ELEMENT UNITNAME (#PCDATA)>
<!ELEMENT NAMEVALUE ( NAME, VALUE, UNITNAME?)>
<!ELEMENT KEYVALUE ( KEY, VALUE, UNITNAME?)>
<!-- Reusable entity specifying NAME, KEY or both elements is valid-->
<!ENTITY % name_key “((NAME, KEY?) | KEY)”>
<!ELEMENT OWNER %name_key;>

<!ELEMENT CATEGORY (%name_key;?, TYPE?, NAMEVALUE*, UPDATE?, METADATA*, COMMENT?)>
<!ATTLIST CATEGORY ACTION (ADD | DELETE | UPDATE |NONE) #REQUIRED>

<!ELEMENT ATTRIBUTE (%name_key;?, OWNER?, LENGTH?, SEARCHABLE?, VISIBILITY?, TYPE?, UNIT*,
UPDATE?, METADATA*, NAMEVALUE*, MULTIVALUE?, COMMENT?)>
<!ATTLIST ATTRIBUTE ACTION (ADD | DELETE | UPDATE | NONE) #REQUIRED>
<!ELEMENT MULTIVALUE (VALUE)+>
<!ELEMENT UNIT (UNITELEMENT+, UPDATE?)>
<!ATTLIST UNIT ACTION (ADD | DELETE | UPDATE) #REQUIRED>
<!ELEMENT UNITELEMENT (%name_key;, MULTIPLIER?, ADDER?)>
<!ATTLIST UNITELEMENT type (BASE | DERIVED) #REQUIRED>
<!ELEMENT MULTIPLIER ( #PCDATA )>
<!ELEMENT ADDER ( #PCDATA )>
```

```
<!ELEMENT NAVIGATION (%name_key;?, OWNER?, TYPE?, NAMEVALUE*, UPDATE?, COMMENT?)>
<!ATTLIST NAVIGATION ACTION (ADD | DELETE | UPDATE |NONE) #REQUIRED>

<!ELEMENT METADATA (NAME?, VALUE?, TYPE?, UPDATE?, COMMENT?)>
<!ATTLIST METADATA ACTION (ADD | DELETE | UPDATE |NONE) #REQUIRED>

<!ELEMENT ITEM (OWNER*, (NAMEVALUE* | KEYVALUE*), UPDATE?, DATAEXTENSION?, METADATA*,
COMMENT?)>
<!ATTLIST ITEM ACTION (ADD | DELETE | UPDATE | MOVE | COPY | NONE) #REQUIRED>

<!ELEMENT DATAEXTENSION (TYPE, NAME?, DATAELEMENT+)>
<!ELEMENT DATAELEMENT (NAMEVALUE*, UPDATE?, COMMENT?)>
<!ATTLIST DATAELEMENT ACTION (ADD | DELETE | UPDATE) #REQUIRED>

<!ELEMENT UPDATE (NAME?, VALUE?, OWNER?, TYPE?, (NAMEVALUE* | KEYVALUE*),
DATAEXTENSION?,UNITELEMENT?, PRICEELEMENT?, SEARCHABLE?, VISIBILITY?)>

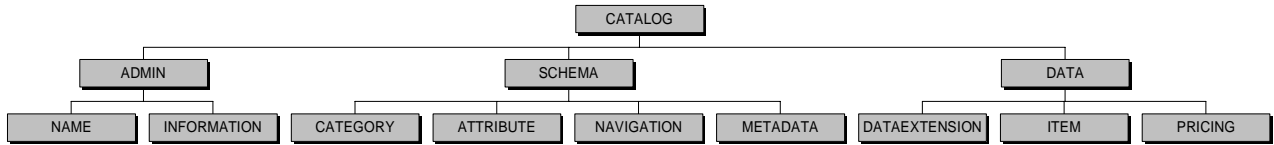
<!ELEMENT PRICING (ITEM, PRICEELEMENT*, UPDATE?, COMMENT?)>
<!ATTLIST PRICING ACTION (ADD|DELETE|UPDATE) #REQUIRED>
<!ELEMENT PRICEELEMENT (SITEPRICE | DISCOUNTPRICE | VOLUMEPRICE)*>
<!ATTLIST PRICEELEMENT currency (#PCDATA) #REQUIRED>
<!ELEMENT SITEPRICE (PRICEVALUE, SITE)>
<!ELEMENT DISCOUNTPRICE ((PRICEVALUE|PRICEDISCOUNT|PERCENTDISCOUNT), SITE?)>
<!ELEMENT VOLUMEPRICE (QUANTITY, (PRICEVALUE|PRICEDISCOUNT|PERCENTDISCOUNT), SITE?)>
<!ELEMENT PRICEVALUE (#PCDATA)>
<!ELEMENT SITE (#PCDATA)>
<!ELEMENT PRICEDISCOUNT (#PCDATA)>
<!ELEMENT PERCENTDISCOUNT (#PCDATA)>
<!ELEMENT QUANTITY (START, END)>
<!ELEMENT START (#PCDATA)>
<!ELEMENT END (#PCDATA)>
]>
```

The DTD is intentionally built with use of the OR and OPTIONAL construct as well as PCDATA. While this makes the data definition somewhat less "type-safe", it provides maximum flexibility. This approach also allows arbitrary ordering for many of the elements. Alternative approaches require a very stringent data definition and severely restrict flexibility. Still other alternatives do not use a DTD for validation.

Catalog information is divided into three main sections:

1. Administrative information <ADMIN>
2. Catalog Schema <SCHEMA>
3. Catalog Data Items <DATA>

The following diagram shows a hierarchical representation of the data type definition.



**Figure 1.** eCX Data Type Definition

## Administrative Information

The admin tag is used to identify the catalog, and provide additional information about the catalog such as date, source, author, title, titlelogo, and domain. This tag is required for catalogs. The following is an example of the admin tag:

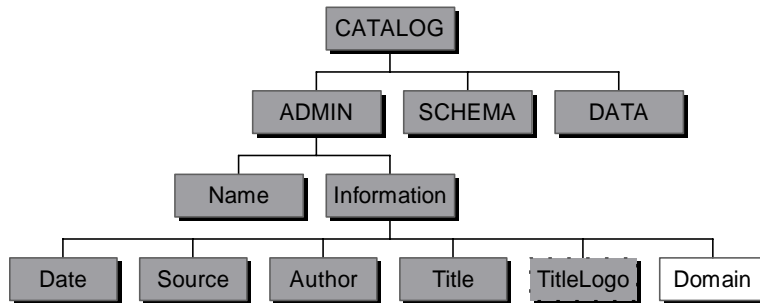
```

<ADMIN>
  <NAME>Business Essentials Reference Catalog</NAME>
  <INFORMATION>
    <DATE>02-FEB-2000</DATE>
    <SOURCE>Requisite Technology Inc.</SOURCE>
    <AUTHOR>eMerge 2.0</AUTHOR>
    <TITLE>BERC</TITLE>
    <TITLELOGO>CompanyLogo.gif</TITLELOGO>
    <DOMAIN>Products</DOMAIN>
  </INFORMATION>
</ADMIN>
  
```

The information contained in this section is source catalog-specific and does not impact the loading of schema or data for other catalogs. The administration information can be used by the catalog load process for a variety of purposes. We recommend the following:

- **Name** is the descriptive name of the catalog.
- **Date** is catalog creation, or extraction, date. We recommend the following format: 03-JAN-1999
- **Source** is the company (content provider, supplier, etc.) that created the XML.
- **Author** is the author of the document. It could contain a tool name and version number or person. (optional tag)
- **Title** represents the name of the catalog.
- **Titlelogo** defines a logo to be associated with the title for content branding. (optional tag)
- **Domain** represents the type of content in the catalog. Some representative values are Products, Services, People, Courses, and other content types. (optional tag)

The following diagram shows a hierarchical diagram of the ADMIN tag.



**Figure 2.** eCX ADMIN Element

## Catalog Structure or Schema

The <SCHEMA> element contains the sub-elements CATEGORY, ATTRIBUTE, NAVIGATION, and METADATA, which make up the set of schema elements. Each schema element has an attribute that specifies the action to perform and sub-elements that specify the actionable data. The available options for the action attribute are ADD, UPDATE, DELETE, and NONE.

The CATEGORY element is used to represent a category as it exists in a catalog. A category is a specifically defined division in a classification that allows you to group classes of objects for organization. The ATTRIBUTE element represents attributes that describe the categories. NAVIGATION elements are used to provide a hierarchical navigation scheme for a catalog. METADATA can be used to distribute additional information about catalog elements. Below are descriptions of each schema element along with examples for clarification. The schema elements need not be in any particular order, however it is recommended that categories be placed before their associated attributes.

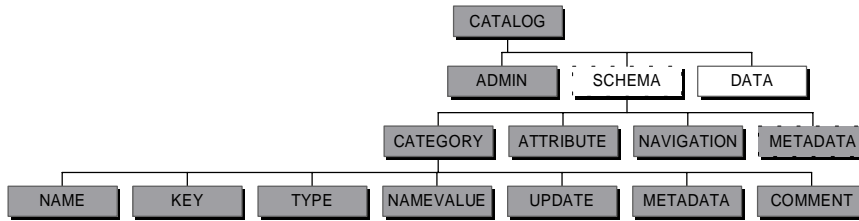
## Category

A category may be identified by name, key, or both. When a key is provided, it is given precedence in identifying a category. Keys are important when the text representation or name of a category is subject to change (e.g., pen is changed to pens). A category may also contain a TYPE, NAMEVALUE, UPDATE, METADATA, or COMMENT element. The category TYPE can be used by the application in order to support the different kinds of categories that might be found in a hierarchical catalog structure. If the type tag is not provided, the application should provide a default type. Application specific information that cannot be represented by the current CATEGORY elements can be represented using the NAMEVALUE element. The METADATA element is used to deliver extended category information. The COMMENT tag is often used to transfer messages between applications, but can be utilized by the application as seen fit.

The following are recommended types to be used for categories:

- Navigation
- Query
- Genus
- Genusnavigation
- Genusquery
- Or other application specific types

The following diagram shows a hierarchical diagram of the CATEGORY schema element.



**Figure 3. eCX CATEGORY Element**

Attribute

An attribute may be identified by a name, a key, or both. When a key is provided, it is given precedence in identifying the attribute. An attribute can also contain OWNER, LENGTH, SEARCHABLE, VISIBILITY, TYPE, UNIT, UPDATE, METADATA, NAMEVALUE, MULTIVALUE, or COMMENT elements. The OWNER element is required when adding attributes, as it associates the attribute with a category. If you specify an owner of "Root", or "0", the attribute will be available for all categories and items. When an attribute has an owner of "Root," this means that the attribute is a root or base attribute. Base attributes are common among all items in a catalog. Examples of a Base attribute are a part number, description, or Stock Keeping Unit (SKU). A local attribute is specific to a particular category (e.g., Ink Color for Pens). The LENGTH element is used to define the maximum length of the values for an attribute. SEARCHABLE is a Boolean attribute that says whether the attributes values should be searchable or not. VISIBILITY controls whether or not the attribute is visible within the application. The TYPE tag for the attribute should not be confused with the category type (recommended attribute types listed below). If the TYPE tag is not provided, the application should determine an appropriate default type. The UNIT element is used to describe an attribute's standard of measure. The UNIT element is made up of UNITELEMENT(s) that can be of type BASE or DERIVED. A BASE UNITELEMENT specifies the default unit of measure and a DERIVED UNITELEMENT specifies an alternate unit of measure and a multiplier or adder that should be applied to the base for obtaining the derived unit of measure value. The METADATA element is used to deliver extended attribute information. Application specific information that cannot be represented by the current ATTRIBUTE elements can be represented using the NAMEVALUE element. The COMMENT tag is often used to transfer messages between applications, but can be utilized by the application as seen fit.

The following are recommended types to be used for attributes:

- String
- Numeric
- Enumerated
- Graphic
- URL
- Date
- Currency
- Currency Unit
- Configurator
- International String

The following diagram shows a hierarchical representation of the ATTRIBUTE schema element.

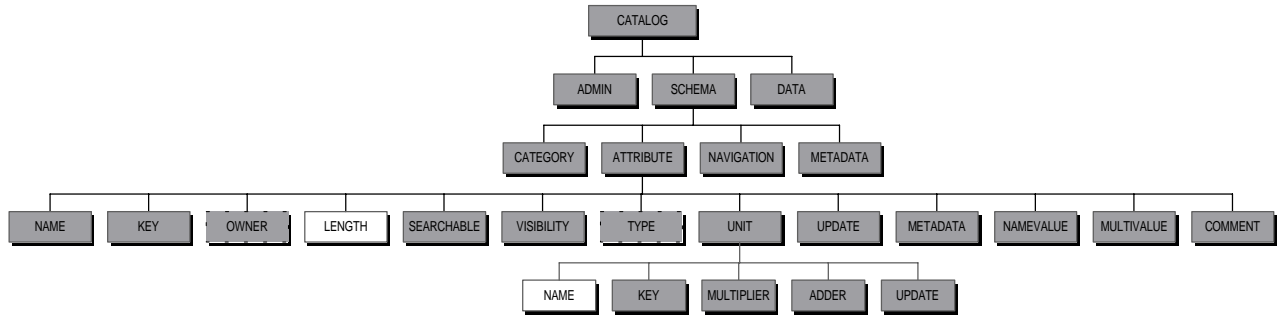


Figure 4. eCX ATTRIBUTE Element

### Navigation

The NAVIGATION element is used to represent hierarchical category structures. Each NAVIGATION element represents a node in the hierarchy. The NAME and/or KEY are used to identify the node of interest, if both are specified, key is given precedence. The OWNER element identifies a NAVIGATION element’s parent node. The TYPE tag for the NAVIGATION element enables support for application-specific kinds of hierarchy nodes (recommended navigation types listed below). If the TYPE tag is not provided, the application should determine an appropriate default type. Application-specific information that cannot be represented by the current NAVIGATION elements can be represented using the NAMEVALUE element. The COMMENT tag is often used to transfer messages between applications, but can be utilized by the application as seen fit.

The following are recommended types to be used for navigation elements:

- Navigation
- Query
- Genus
- Genusnavigation
- Genusquery
- Other application-specific types

The following diagram shows a hierarchical representation of the NAVIGATION schema element.

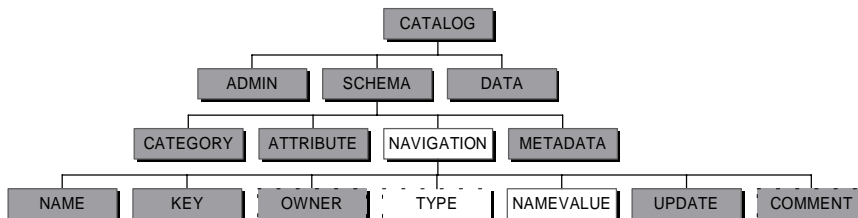
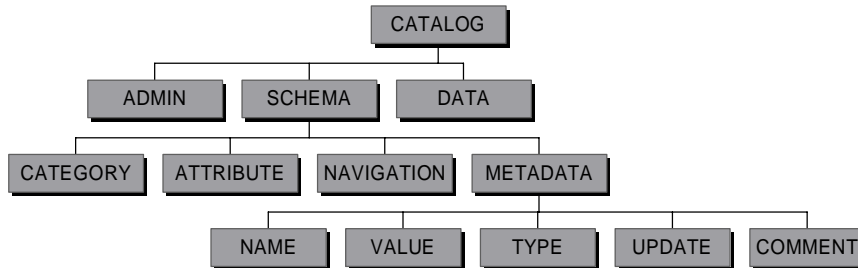


Figure 5. eCX NAVIGATION Element

Metadata

The METADATA element is used to transfer additional information about catalog elements. Typically the METADATA element is contained within one of the other catalog elements, however the concept of metadata is broad enough to warrant a standalone element that enables applications to define proprietary catalog information exchange and manipulation of metadata across catalog elements. Metadata information is identified by the NAME element with the associated VALUE element used as a container for the metadata information. The TYPE element enables metadata to be grouped or associated with catalog elements and information. The COMMENT tag is often used to transfer messages between applications, but can be utilized by the application as seen fit.

The following diagram shows a hierarchical representation of the METADATA schema element.



**Figure 6.** eCX METADATA Element

Examples: Add Category or Attribute

Catalog schema is added by specifying an action of ADD for the different schema elements. The sub-elements of a schema element with an action of ADD are used to identify the data for the new schema element. The flexibility of the DTD makes it possible to specify as few or as many sub-elements as needed. This enables the application to provide control over default values and which values are required.

```

<SCHEMA>
  <!--Add a category specifying the name and key, using the default type -->
  <CATEGORY ACTION="ADD">
    <NAME>Pen Gift Set</NAME>
    <KEY>Pen Gift Sets</KEY>
  </CATEGORY>

  <!-- Add a category specifying the name letting the application determine the key and type -->
  <CATEGORY ACTION="ADD">
    <NAME>Computer</NAME>
  </CATEGORY>

  <!-- Add a category specifying the key and type letting the application determine the name -->
  <CATEGORY ACTION="ADD">
    <KEY>Laptop Computer</KEY>
    <TYPE>GENUS</TYPE>
    <METADATA ACTION="ADD">
      <NAME>Supplier Category</NAME>
      <VALUE>Laptop's</VALUE>
    </METADATA>
  </CATEGORY>
  
```

<!-- Add an attribute called Product Shipping Weight to the Pen Gift Set category. Pound is the base unit of measure and gram is a derived with its conversion specified-->

```
<ATTRIBUTE ACTION="ADD">
  <NAME>Product Shipping Weight</NAME>
  <OWNER>
    <NAME>Pen Gift Set</NAME>
  </OWNER>
  <LENGTH>LENGTH</LENGTH>
  <SEARCHABLE>YES</SEARCHABLE>
  <VISIBILITY>YES</VISIBILITY>
  <TYPE>String</TYPE>
  <UNIT ACTION="ADD">
    <UNITELEMENT type="BASE">
      <NAME>Pounds</NAME>
    </UNITELEMENT>
    <UNITELEMENT type="DERIVED">
      <NAME> Gram </NAME>
      <MULTIPLIER>453.5</MULTIPLIER>
      <ADDER> 0</ADDER>
    </UNITELEMENT>
  </UNIT>
</ATTRIBUTE>
```

<!-- Add a node to the category hierarchy, using the NAMEVALUE element to represent application specific information -->

```
<NAVIGATION ACTION="ADD">
  <NAME>Office Supplies</NAME>
  <KEY>Office Supplies</KEY>
  <OWNER>
    <NAME>Business Needs</NAME>
  </OWNER>
  <TYPE>Navigation</TYPE>
  <NAMEVALUE>
    <NAME>SEQUENCE</NAME>
    <VALUE>0</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>DESCRIPTION</NAME>
    <VALUE>Essential items for the office</VALUE>
  </NAMEVALUE>
</NAVIGATION>
```

<!-- Add a node to the hierarchy that lies below the Office Supplies category, using the NAMEVALUE element to represent application specific information -->

```
<NAVIGATION ACTION="ADD">
  <NAME>Pen Gift Set</NAME>
  <KEY>Pen Gift Sets</KEY>
  <OWNER>
    <NAME>Office Supplies</NAME>
    <KEY>Office Supplies</KEY>
  </OWNER>
  <TYPE>Genus</TYPE>
  <NAMEVALUE>
    <NAME>SEQUENCE</NAME>
    <VALUE>0</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
</NAMEVALUE>
```

```

        <NAME>DESCRIPTION</NAME>
        <VALUE>All pen gift sets available for purchase</VALUE>
    </NAMEVALUE>
</NAVIGATION>
</SCHEMA>

```

#### Examples: Delete Category or Attribute

Catalog schema is deleted by specifying an action of DELETE for the different schema elements. The sub-elements of a schema element with an action of DELETE are used to identify what pieces of the catalog schema will be deleted. Specifying more or fewer sub-elements constrains or broadens what pieces of the catalog will be deleted.

```

<SCHEMA>
  <!-- Delete the category with a name of Pen Gift Set -->
  <CATEGORY ACTION="DELETE">
    <NAME>Pen Gift Set</NAME>
  </CATEGORY>

  <!-- Delete the category with a key of Computer -->
  <CATEGORY ACTION="DELETE">
    <KEY>Computer</KEY>
  </CATEGORY>

  <!-- Delete the Barrel Color attribute for the Pens category -->
  <ATTRIBUTE ACTION="DELETE">
    <NAME>Barrel Color</NAME>
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
  </ATTRIBUTE>

  <!-- Delete all the attributes that are hidden within the application -->
  <ATTRIBUTE ACTION="DELETE">
    <HIDDEN>YES</HIDDEN>
  </ATTRIBUTE>

  <!-- Delete a node from the hierarchy that lies below the Office Supplies category -->
  <NAVIGATION ACTION="DELETE">
    <NAME>Pen Gift Set</NAME>
    <OWNER>
      <NAME>Office Supplies</NAME>
    </OWNER>
  </NAVIGATION>
</SCHEMA>

```

## Examples: Update a Category or Attribute

Catalog schema is updated by specifying an action of UPDATE for the different schema elements. Sub-elements, outside of the UPDATE element, within a schema element having an action of UPDATE, are used to identify what pieces of the catalog schema will be updated. Specifying more or fewer sub-elements controls what pieces of the catalog will be updated. Elements contained within the UPDATE element are used to specify the new values for the catalog schema.

```

<SCHEMA>
  <!-- Rename the category Pen Gifts Set to Pen Gift Sets -->
  <CATEGORY ACTION="UPDATE" >
    <NAME>Pen Gift Set</NAME>
    <UPDATE>
      <NAME>Pen Gift Sets</NAME>
    </UPDATE>
  </CATEGORY>

  <!-- Rename the category with the key Comp Motors to Compressor Motors -->
  <CATEGORY ACTION="UPDATE">
    <KEY>Comp Motors</KEY>
    <UPDATE>
      <NAME>Compressor Motors</NAME>
    </UPDATE>
  </CATEGORY>

  <!-- Update the metadata for a category -->
  <CATEGORY ACTION="UPDATE">
    <KEY>Laptop Computer</KEY>
    <METADATA ACTION="UPDATE">
      <NAME>Supplier Category</NAME>
      <VALUE>Laptop's</VALUE>
    <UPDATE>
      <NAME>Supplier Categories</NAME>
      <VALUE>Laptop's, Portable Computers</VALUE>
    </UPDATE>
  </METADATA>
  </CATEGORY>

  <!-- Rename the Ink Color attribute for the Pens category to Ink Colors, and make the attribute
  searchable within the application -->
  <ATTRIBUTE ACTION="UPDATE">
    <NAME>Ink Color</NAME>
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
    <UPDATE>
      <NAME>Ink Colors</NAME>
      <SEARCHABLE>YES</SEARCHABLE>
    </UPDATE>
  </ATTRIBUTE>

  <!-- Delete the Gram unit of measure from the Product Shipping Weight attribute -->
  <ATTRIBUTE ACTION="UPDATE">
    <NAME>Product Shipping Weight</NAME>
    <UNIT ACTION="DELETE">
      <UNITELEMENT type="DERIVED">

```

```
        <NAME>Gram</NAME>
      </UNITELEMENT>
    </UNIT>
  </ATTRIBUTE>
</SCHEMA>
```

## Catalog Data

The <DATA> element contains the sub-elements ITEM, DATAEXTENSION, and PRICING, which make up the set of data elements. The ITEM and PRICING elements both have an attribute that specifies the action to perform and sub-elements that specify the actionable data, where as the DATAEXTENSION element has DATAELEMENT sub-elements that specify the action. Each element supports the ADD, UPDATE, DELETE, and NONE options for the action attribute.

The ITEM element is used to represent an item, as it exists in the catalog. The item element can be used to manipulate all aspects of an item within a catalog. The DATAEXTENSION element is used to support application-specific extensions for catalog data input. The PRICING element allows for sophisticated pricing schemes within a catalog. Below are descriptions of each data element along with examples for clarification.

### ITEM

Catalog Data items can be added to a category, deleted, and modified. The OWNER element(s) within an ITEM element are used to identify the category(ies) that the item belongs to. An item can also contain NAMEVALUE or KEYVALUE, UPDATE, and DATAEXTENSION tags. The NAMEVALUE or KEYVALUE tags are used to specify the attribute values for an item, and optionally the unit of measure for the value. NAMEVALUE(s) are used when attributes are to be identified by name and KEYVALUE(s) are used when attributes are to be identified by key. The use of the name/value pair **eliminates the need for catalog vendors or XML "standards" to "agree" on specific tags**. The responsibility for the interchange is left to the catalog loader vendor and not the community of content providers, suppliers, and buyers. The UPDATE tag is used to contain new item information when an action of UPDATE was specified for the item. A DATAEXTENSION tag within an ITEM tag enables item associations or extended item information to be represented. Some examples might be product associations via a query linkage (i.e., ink refills for a pen), or manufacturer home page links for items.

Catalog Data Items can also be moved between categories or copied to new categories. To support moving and copying of items, the action attribute for items has the additional options of MOVE and COPY. When performing a move or a copy, the data within the UPDATE element identifies the new category and the mapping of the old category attributes to the new category attributes. The attribute mapping is specified using the NAMEVALUE or KEYVALUE elements. The NAME or KEY, in NAMEVALUE or KEYVALUE respectively, represents the name or key of an attribute in the new category and the value represents the name or key of an attribute in the current category. The data outside the UPDATE element is used to identify the item(s) to move or copy.

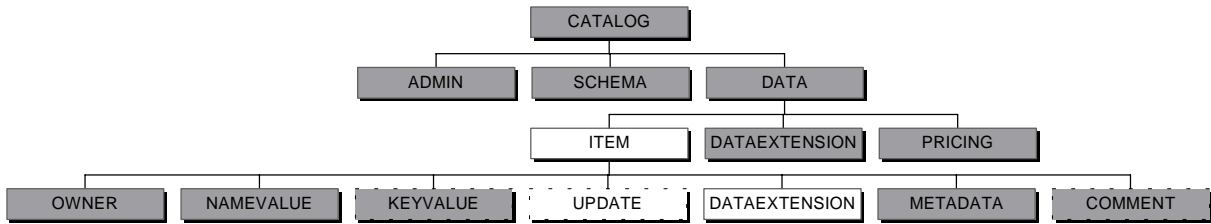
Example: Copy Items

```
<DATA>
  <!-- Copy the item with a supplier part num of 123456 into the Fine Writing Pens category. Map
  the value for the Color and Point attributes in the Pens category to Ink Color and Point Style in the
  Fine Writing Pens category-->
  <ITEM ACTION="COPY">
    <OWNER><NAME>Pens</NAME></OWNER>
    <KEYVALUE>
      <KEY>Sup Part Num</KEY>
      <VALUE>123456</VALUE>
    </KEYVALUE>
    <UPDATE>
      <OWNER><KEY>Fine Writing Pens</KEY></OWNER>
      <KEYVALUE>
        <KEY>Ink Color</KEY>
        <VALUE>Color</VALUE>
      </KEYVALUE>
      <KEYVALUE>
        <KEY>Point Style</KEY>
        <VALUE>Point</VALUE>
      </KEYVALUE>
    </UPDATE>
  </ITEM>
</DATA>
```

Example: Move Items

```
<DATA>
  <!-- Move all items in the Pens to the Markers category. Map the value for the Color and Point
  attributes in the Pens category to Ink Color and Point Style in the Fine Writing Pens category -->
  <ITEM ACTION="MOVE">
    <OWNER><NAME>Pens</NAME><OWNER>
    <UPDATE>
      <OWNER><KEY>Fine Writing Pens</KEY><OWNER>
      <KEYVALUE>
        <KEY>Ink Color</KEY>
        <VALUE>Color</VALUE>
      </KEYVALUE>
      <KEYVALUE>
        <KEY>Point Style</KEY>
        <VALUE>Point</VALUE>
      </KEYVALUE>
    </UPDATE>
  </ITEM>
</DATA>
```

The following diagram shows a hierarchical representation of the ITEM element.



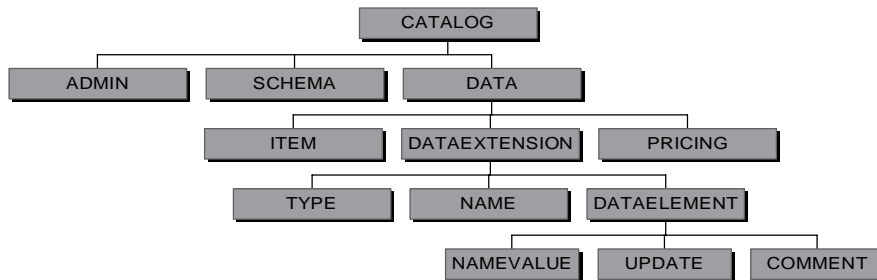
**Figure 7. eCX ITEM Element**

## Data Extensions

There arises the need in catalog data exchange to provide a mechanism to support extensions and application-specific data input requirements. One approach would be to invent tags for each one of these options. This specification provides the mechanism by allowing a data extension tag that allows for application unique data.

The TYPE element within the DATAEXTENSION element is used for grouping a set of data extension information. A data extension also contains a NAME and DATAELEMENT tags. The NAME element is used to provide further grouping information, and the DATAELEMENT tags are used to provide the application specific data input. In the item description above, the Association type data extension was described as a way for associating items with extended information. Another example of a data extension might be of type “Table” that would allow for transferring table-based application specific data. If the type was “Table”, the name could be used to specify a table name and the data elements could be used to contain row information for the table. Examples of this type of data extension are provided in the DATA element example sections.

The following diagram shows a hierarchical representation of the DATAEXTENSION element.

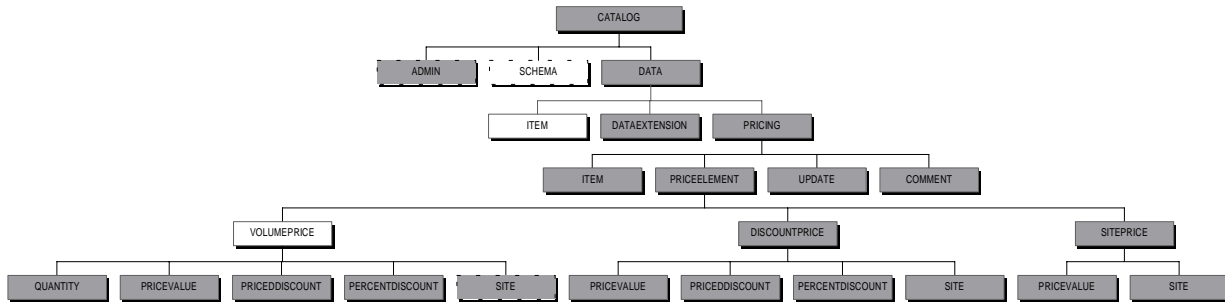


**Figure 8. eCX DATAEXTENSION Element**

## Pricing

The PRICING element is used to provide sophisticated pricing schemes for a catalog. A PRICING element can contain an ITEM, PRICEELEMENT, and UPDATE tags. The ITEM element is used to identify the items the pricing information is for. By reviewing the ITEM specification it can be seen that this allows for pricing to be applied on an item, by item basis or in mass. The PRICEELEMENT tag contains the details of the pricing scheme(s) of the item(s) for the currency specified in the currency attribute. The pricing schemes supported are volume pricing, discount pricing, and site specific pricing. Both volume pricing and discount pricing can be specified on a site-by-site basis.

The following diagram shows a hierarchical representation of the PRICING element.



**Figure 9.** eCX PRICING Element

Example: Add Items, Data Extensions, Pricing

Data elements are added by specifying an action of ADD for the different data elements. The sub-elements of a data element with an action of ADD are used to identify the data for the new data element. The flexibility of the DTD makes it possible to specify as few or as many sub-elements as needed. This enables the application to provide control over default values and which values are required.

```

<DATA>
  <!-- Add an item to the Pens category -->
  <ITEM ACTION="ADD">
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
    <NAMEVALUE>
      <NAME>Mfg Name</NAME>
      <VALUE>Bic</VALUE>
    </NAMEVALUE>
    <NAMEVALUE>
      <NAME>Price</NAME>
      <VALUE>1.55</VALUE>
    </NAMEVALUE>
    <NAMEVALUE>
      <NAME>Barrel Color</NAME>
      <VALUE>Blue</VALUE>
    </NAMEVALUE>
  </ITEM>
  <!-- Add an item to the Pens category specifying both accessory and manufacturer home page
  extended information for the item -->
  <ITEM ACTION="ADD">
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
    <NAMEVALUE>
      <NAME>Mfg Name</NAME>
      <VALUE>Bic</VALUE>
    </NAMEVALUE>
    <DATAEXTENSION>
      <TYPE>Association</TYPE>
  </ITEM>
  
```

```

<DATAELEMENT ACTION="ADD">
  <NAMEVALUE>
    <NAME>TYPE</NAME>
    <VALUE>BugsEye</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>QUERY</NAME>
    <VALUE>Ink Refill</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>COLUMN_NAME</NAME>
    <VALUE>Accessories</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>GRAPHIC</NAME>
    <VALUE>access.gif</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>LABEL</NAME>
    <VALUE>Accessories Text</VALUE>
  </NAMEVALUE>
</DATAELEMENT>
<DATAELEMENT ACTION="ADD">
  <NAMEVALUE>
    <NAME>TYPE</NAME>
    <VALUE>URL</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>QUERY</NAME>
    <VALUE>http://www.bic.com/</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>COLUMN_NAME</NAME>
    <VALUE>Accessories</VALUE>
  </NAMEVALUE>
  <NAMEVALUE>
    <NAME>LABEL</NAME>
    <VALUE>Mfg Home Page</VALUE>
  </NAMEVALUE>
</DATAELEMENT>
</DATAEXTENSION>
</ITEM>
<!-- The data extension below demonstrates how an application can use data extensions to
expand the specification to support the unique needs of the application.
Add a row to a configuration table in the application software, telling it to enable its shopping cart--
>
<DATAEXTENSION>
  <TYPE>Table</TYPE>
  <NAME>configuration</NAME>
  <DATAELEMENT ACTION="ADD">
    <NAMEVALUE>
      <NAME>value</NAME>
      <VALUE>True</VALUE>
    </NAMEVALUE>
    <NAMEVALUE>
      <NAME>name</NAME>

```

```

                <VALUE>ShoppingCart</VALUE>
            </NAMEVALUE>
        </DATAELEMENT>
    </DATAEXTENSION>
    <!-- Create volume pricing for item PN1234 at the Boulder site. Volume of 10 or
    less cost $2.00, volumes 11-20 cost $1.50 -->
    <PRICING ACTION="ADD">
        <ITEM ACTION="NONE">
            <NAMEVALUE>
                <NAME>Sup Part Num</NAME>
                <VALUE>PN1234</VALUE>
            </NAMEVALUE>
        </ITEM>
        <PRICEELEMENT currency="USD">
            <VOLUMEPRICE>
                <QUANTITY>
                    <START>1</START>
                    <END>10</END>
                </QUANTITY>
                <PRICEVALUE>2.0</PRICEVALUE>
                <SITE>Boulder</SITE>
            </VOLUMEPRICE>
            <VOLUMEPRICE>
                <QUANTITY>
                    <START>11</START>
                    <END>20</END>
                </QUANTITY>
                <PERCENT>15</PERCENT>
                <SITE>Boulder</SITE>
            </VOLUMEPRICE>
        </PRICEELEMENT>
    </PRICING>
    <!-- Create a discount price for Joe's Office Supplies where purchases at the Boulder site get 10
    dollars off and all other purchases get a 15% discount -->
    <PRICING ACTION="ADD">
        <ITEM ACTION="NONE">
            <NAMEVALUE>
                <NAME>Sup Name</NAME>
                <VALUE>Joes Office Supplies</VALUE>
            </NAMEVALUE>
        </ITEM>
        <PRICEELEMENT currency="USD">
            <DISCOUNTPRICE>
                <PRICEVALUE>10</PRICEVALUE>
                <SITE>Boulder</SITE>
            </DISCOUNTPRICE>
        </PRICEELEMENT>
        <PRICEELEMENT currency="USD">
            <DISCOUNTPRICE>
                <PERCENTDISCOUNT>15</PERCENTDISCOUNT>
            </DISCOUNTPRICE>
        </PRICEELEMENT>
    </PRICING>
</DATA>

```

## Example: Delete Items, Data Extensions, Pricing

Catalog data is deleted by specifying an action of DELETE for the different data elements. The sub-elements of a data element with an action of DELETE are used to identify what pieces of the catalog data will be deleted. Specifying more or fewer sub-elements controls what data in the catalog will be deleted.

```

<DATA>
  <!-- Delete all items in the Pens category with a Mfg Name of Bic and a Barrel Color of Blue-->
  <ITEM ACTION="DELETE">
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
    <NAMEVALUE>
      <NAME>Mfg Name</NAME>
      <VALUE>Bic</VALUE>
    </NAMEVALUE>
    <NAMEVALUE>
      <NAME>Barrel Color</NAME>
      <VALUE>Blue</VALUE>
    </NAMEVALUE>
  </ITEM>
  <!-- The data extension below demonstrates how an application can use data extensions to
  expand the specification to support the unique needs of the application.
  Delete the Denver configuration from the application-->
  <DATAEXTENSION>
    <TYPE>Table</TYPE>
    <NAME>configuration</NAME>
    <DATAELEMENT ACTION="DELETE">
      <NAMEVALUE>
        <NAME>config_name</NAME>
        <VALUE>Denver</VALUE>
      </NAMEVALUE>
    </DATAELEMENT>
  </DATAEXTENSION>
  <!-- Remove extended pricing for PN1234, so only list price is available -->
  <PRICING ACTION="DELETE">
    <ITEM ACTION="NONE">
      <NAMEVALUE>
        <NAME>Sup Part Num</NAME>
        <VALUE>PN1234</VALUE>
      </NAMEVALUE>
    </ITEM>
  </PRICING>
  <!-- Remove $10 dollar discount from the Boulder site for Joe's Office Supplies-->
  <PRICING ACTION="DELETE">
    <ITEM ACTION="NONE">
      <NAMEVALUE>
        <NAME>Sup Name</NAME>
        <VALUE>Joe's Office Supplies</VALUE>
      </NAMEVALUE>
    </ITEM>
    <PRICEELEMENT currency="USD">
      <DISCOUNTPRICE>
        <PRICEVALUE>10</PRICEVALUE>
        <SITE>Boulder</SITE>
      </DISCOUNTPRICE>
    </PRICEELEMENT>
  </PRICING>

```

```

        </DISCOUNTPRICE>
    </PRICEELEMENT>
</PRICING>
</DATA>

```

Example: Update Items, Data Extensions, Pricing

Catalog data is updated by specifying an action of UPDATE for the different data elements. Sub-elements, outside of the UPDATE element, within a data element having an action of UPDATE are used to identify what catalog data will be updated. Specifying more or fewer sub-elements controls what catalog data will be updated. Elements contained within the UPDATE element are used to specify the new values for the catalog data.

```

<DATA>
  <!-- Change the Mfg Name to Bic Inc. and the Barrel Color to Red for the item in the pens category
  with a Mfg Name of Bic and a Sup Part Num of B145C7 -->
  <ITEM ACTION="UPDATE">
    <OWNER>
      <NAME>Pens</NAME>
    </OWNER>
    <KEYVALUE>
      <KEY>Mfg Name</KEY>
      <VALUE>Bic</VALUE>
    </KEYVALUE>
    <KEYVALUE>
      <KEY>Sup Part Num</KEY>
      <VALUE>B145C7</VALUE>
    </KEYVALUE>
    <UPDATE>
      <KEYVALUE>
        <KEY>Mfg Name</KEY>
        <VALUE>Bic Inc.</VALUE>
      </KEYVALUE>
      <KEYVALUE>
        <KEY>Barrel Color</KEY>
        <VALUE>Red</VALUE>
      </KEYVALUE>
    </UPDATE>
  </ITEM>
  <!-- Change the manufacturer name from to Bic Inc. for all items whose current manufacturer
  name is Bic -->
  <ITEM ACTION="UPDATE">
    <NAMEVALUE>
      <NAME>Mfg Name</NAME>
      <VALUE>Bic</VALUE>
    </NAMEVALUE>
    <UPDATE>
      <NAMEVALUE>
        <NAME>Mfg Name</NAME>
        <VALUE>Bic Inc.</VALUE>
      </NAMEVALUE>
    </UPDATE>
  </ITEM>

```

<!-- The data extension below demonstrates how an application can use data extensions to expand the specification to support the unique needs of the application. Rename the Denver configuration in the application to Metro Denver-->

```

<DATAEXTENSION>
  <TYPE>Table</TYPE>
  <NAME>configuration</NAME>
  <DATAELEMENT ACTION="UPDATE">
    <NAMEVALUE>
      <NAME>config_name</NAME>
      <VALUE>Denver</VALUE>
    </NAMEVALUE>
    <UPDATE>
      <NAME>config_name</NAME>
      <VALUE>Metro Denver</VALUE>
    </UPDATE>
  </DATAELEMENT>
</DATAEXTENSION>
<!-- Update discount price to 20% for Joe's Office Supplies-->
<PRICING ACTION="UPDATE">
  <ITEM ACTION="NONE">
    <NAMEVALUE>
      <NAME>Sup Name</NAME>
      <VALUE>Joe's Office Supplies</VALUE>
    </NAMEVALUE>
  </ITEM>
  <UPDATE>
    <PRICEELEMENT>
      <DISCOUNTPRICE>
        <PERCENTDISCOUNT>20</PERCENTDISCOUNT>
      </DISCOUNTPRICE>
    </PRICEELEMENT>
  </UPDATE>
</PRICING>
<!-- Update site discount for Boulder to be 20 dollars for Joe's Office Supplies-->
<PRICING ACTION="UPDATE">
  <ITEM ACTION="NONE">
    <NAMEVALUE>
      <NAME>Sup Name</NAME>
      <VALUE>Joe's Office Supplies</VALUE>
    </NAMEVALUE>
  </ITEM>
  <UPDATE>
    <PRICEELEMENT currency="USD">
      <DISCOUNTPRICE>
        <PRICEVALUE>20</PRICEVALUE>
        <SITE>Boulder</SITE>
      </DISCOUNTPRICE>
    </PRICEELEMENT>
  </UPDATE>
</PRICING>
</DATA>

```

## Special Character Handling

Special characters and non-XML markup must be escaped for XML parsers and catalog loaders to function correctly. Specifically, the & and < characters must be escaped with a CDATA tag.

<![CDATA[ **your data here** ]]> inserted in any tag for special characters.

```
<SCHEMA>
  <CATEGORY ACTION="DELETE">
    <NAME><![CDATA[Pen & Pencil Gifts Sets]]></NAME>
  </CATEGORY>
</SCHEMA>
```

## International Characters

This specification allows the setting of default languages by use of the xml:lang attribute on the CATALOG element. Multiple languages can be contained in this specification by the addition of xml:lang to the NAME and VALUE elements which contain the actual data to be described.

This language encoding follows the ISO standards 639, 3161 for codes and sub codes. This is based on the following language from the "Extensible Markup Language (XML) 1.0" W3C Recommendation 10-February-1998 (section 2-12). It states:

*The intent declared with xml:lang is considered to apply to all attributes and content of the element where it is specified, unless overridden with an instance of xml:lang on another element within that content.*

The following is a table from the XML 1.0 specification stating "how" language identification is specified.

### Language Identification

LanguageID	::=	Langcode ( - Subcode)*
Langcode	::=	ISO639Code   langCode   UserCode
ISO639Code	::=	( [a-z]   [A-Z] ) ( [a-z]   [A-Z] )
IanaCode	::=	( 'i'   'I' ) '-' ( [a-z]   [A-Z] )+
UserCode	::=	( 'x'   'X' ) '-' ( [a-z]   [A-Z] )+
SubCode	::=	( [a-z]   [A-Z] )+

Here's an example where language is specified for the entire catalog where the language = US English and the country is United States:

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
<CATALOG xml:lang="en-us">
```

We recommend that you check with your XML parser and its documentation on how it handles international characters.

## Conclusion

The eCatalog XML specification is currently used by Requisite Technology and its partners to exchange catalog structure and electronic content. It is the intent of this XML specification to promote an open, catalog interoperability standard. The metadata nature of this specification will allow eCatalog XML to incorporate many and varied XML "standards" emerging in electronic commerce and provide true and complete catalog interoperability and content exchange.

Please direct all questions and comments to Requisite Technology at [ecx\\_question@Requisite.com](mailto:ecx_question@Requisite.com).