opentext™

OpenText™ Physical Objects

Label Design Guide

This guide provides information about generating labels and formatting label content for Physical Objects.

LLESPOB160204-RGD-EN-1

OpenText™ Physical Objects
Label Design Guide
LLESPOB160204-RGD-EN-1
Rev.: 2018-Mar-07

This documentation has been created for software version 16.2.4.
It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at https://knowledge.opentext.com.

Open Text Corporation

275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677
Support: https://support.opentext.com
For more information, visit https://www.opentext.com

Table of Contents

Chapter 1

Generating Labels for Physical Objects and Locators

Physical Objects gives users the ability to generate labels for physical objects and Locators. The content and format of these labels, as well as the dimensions of the label stock, are defined using XML stored in the Content Server database.

These labels provide support for:

- Text
- Bar codes
- Color coding
- Rectangles (with or without rounded corners)
- Lines

1.1 The XML Structure

This section details the XML structure for the label stock and the label definition.

In Content Server, a label is defined as a combination of one or more blocks that is typically repeated one or more times on a sheet of label stock. A label block is the portion of the label that is peeled off and applied to the physical object.

See Figures 1-1 and 1-2 for a graphical representation of labels and label blocks.

1.1.1 Label Stock Definitions

The label stock definition portion of the XML file defines the height and width of the label stock page, the height and width of the label block, and the coordinates (for example, top left corner) of each block that will appear on the page.

Figure 1-1 represents a label stock page (8.5" x 11") that has four label blocks positioned on it. These blocks make up two labels, each containing a large and small block.

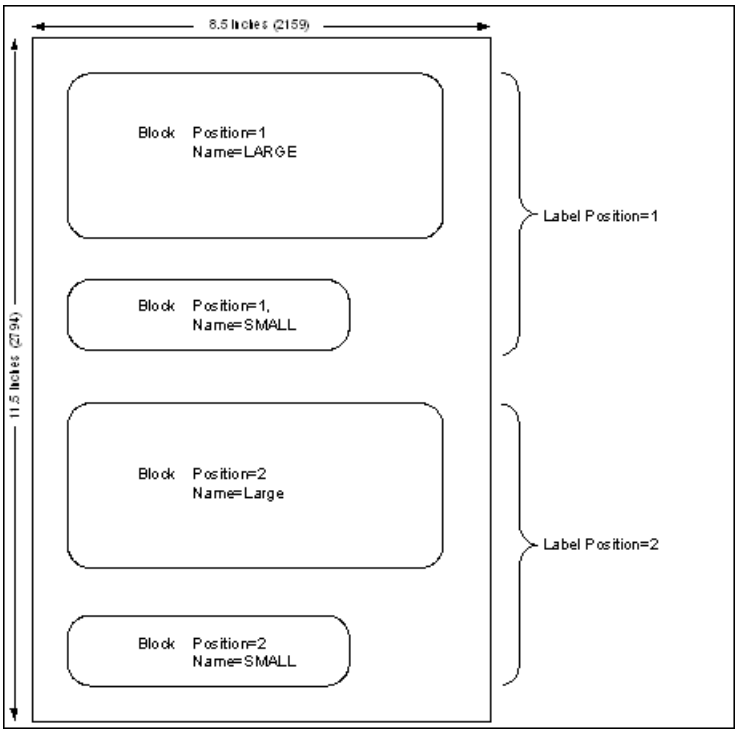


Figure 1-1: Label stock containing two labels comprised of four blocks

```
<LABELSTOCKDEFINITION NAME="2LABELS4BLOCKS" HEIGHT="2794" WIDTH="2159"
                      ORIENTATION ="PORTRAIT">

    <LABEL POSITION="1">
      <BLOCKPOSITION TOP="95" LEFT="65" NAME="LARGE"/>
      <BLOCKPOSITION TOP="940" LEFT="65" NAME="SMALL"/>
    </LABEL>

    <LABEL POSITION="2">
      <BLOCKPOSITION TOP="1489" LEFT="65" NAME="LARGE"/>
      <BLOCKPOSITION TOP="2335" LEFT="65" NAME="SMALL"/>
    </LABEL>

    <BLOCKSIZE NAME="LARGE" HEIGHT="762" WIDTH="2032"/>
    <BLOCKSIZE NAME="SMALL" HEIGHT="213" WIDTH="1397"/>

</LABELSTOCKDEFINITION>
```

Figure 1-2: Sample LabelStockDefinition for Figure 1-1

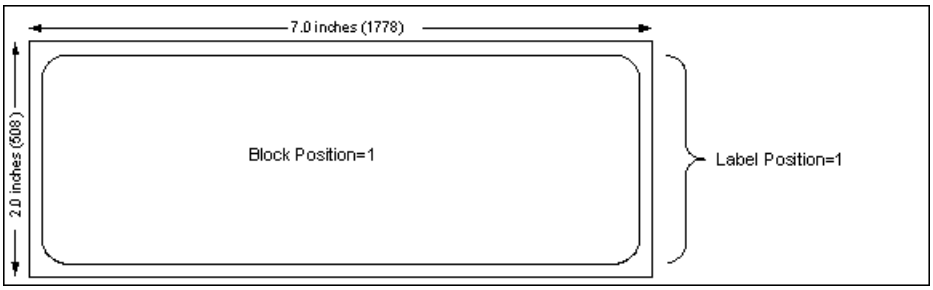


Figure 1-3: Label stock containing one label with one block

```
<LABELSTOCKDEFINITION NAME="7X2LABEL" HEIGHT="540" WIDTH="1805"
                      ORIENTATION ="PORTRAIT">

  <LABEL POSITION="1">
    <BLOCKPOSITION TOP="15" LEFT="10" NAME="LABEL"/>
  </LABEL>

</LABELSTOCKDEFINITION>
```

Figure 1-4: Sample LabelStockDefinition for Figure 1-c

<LABELSTOCKDEFINITION>

For every different label stock, there must be a `<LABELSTOCKDEFINITION>` element in the XML file. This element contains the following attributes:

Table 1-1: LABELSTOCKDEFINITION attributes

Attribute	Definition
NAME	A unique identifier that describes the specific label stock definition For example, `7X2LABEL` or `2LABELS4BLOCKS`
HEIGHT	The height of the label measured in 1/10 of a millimeter
WIDTH	The width of the label measured in 1/10 of a millimeter
ORIENTATION	The orientation of the label when printing

<LABEL>

Within each <LABELSTOCKDEFINITION>, you must insert a <LABEL> element for each label that appears on the stock. This element contains the following attribute.

Table 1-2: LABEL attributes

Attribute	Definition
POSITION	Indicates the order that the label blocks appear on the label stock from top to bottom

<BLOCKPOSITION>

For each label, you must use the <BLOCKPOSITION> element to define the block coordinates (the top left corner of each block found in the label) in relation to the label stock. This element contains the following attributes:

Table 1-3: BLOCKPOSITION attributes

Attributes	Definition
NAME	A unique identifier that describes the specific label block For example, `Large` or `LargeBlock`
TOP	The measured distance from the top of the label stock to the beginning of the block. Measured in 1/10 of a millimeter
LEFT	The measured distance from the left edge of the label stock to the beginning of the block. Measured in 1/10 of a millimeter

<BLOCKSIZE>

For each label, you must use the <BLOCKSIZE> element to define the size of the block. This element contains the following attributes:

Table 1-4: BLOCKSIZE attributes

Attribute	Definition
NAME	A reference to the name of the block the measurement applies to. The name must match a name used in a <BLOCKPOSITION> element.
HEIGHT	The height of the label block measured in 1/10 of a millimeter
WIDTH	The width of the label block measured in 1/10 of a millimeter

1.1.2 Label Definitions

The label definition portion of the XML file defines all the characteristics of the label, including content, text, color, bar codes, and lines. Because labels differ from client to client, there are many optional XML tags that can be inserted into the definition file. The following describes how to use all the tags that configure a label.

<LABELDEFINITION>

The <LABELDEFINITION> element is a required element. It is the top-level container element that defines all the characteristics of the label. This element contains the following attributes:

Table 1-5: LABELDEFINITION attributes

Attributes	Definition
LABELSTOCKDEFINITION	The label stock used for this label
NAME	The name of the label that will be displayed in Content Server

<BLOCKDEFINITION>

The <BLOCKDEFINITION> element is a required element. It is the container element that holds the label information for a specific block in the label. You must have a <BLOCKDEFINITION> element for every block defined in your label stock definition. For example, if the label stock definition contains two blocks, you require two <BLOCKDEFINITION> elements.

Note: There is no verification process to ensure you have the proper number of blocks. If you do not include a <BLOCKDEFINITION>, that block will be left blank when the label is printed.

This element contains the following attribute:

Table 1-6: BLOCKDEFINITION attributes

Attribute	Definition
NAME	The name of the block that you are specifying the content for. It must be a reference to a name of a block defined using the <BLOCKPOSITION> element in the label stock definition.

<TEXT>

The <TEXT> element is an optional element that specifies the location, content, and format of text on the label. This element contains the following attributes:

Table 1-7: TEXT attributes

Attribute	Definition
TOP	The measured distance from the top of the block definition to the beginning of the text. Measured in 1/10 of a millimeter
LEFT	The measured distance from the left of the block definition to the beginning of the text. Measured in 1/10 of a millimeter
HEIGHT	The height of the text box that will contain the text. Measured in 1/10 of a millimeter
WIDTH	The width of the text box that will contain the text. Measured in 1/10 of a millimeter
CONTENT	The content of the text. Can be a literal value, a database field, or a combination of both. For more information, see"Label Content" on page 17.
FONT	The font of the text content
FONTSIZE	The font size of the text content
COLOR	The color of the text content
BOLD	Boolean value. The text is bolded when the value is set to 1.
ITALIC	Boolean value. The text is italicized when the value is set to 1.
UNDERLINE	Boolean value. The text is underlined when the value is set to 1.
ROTATION	A numerical value representing a clockwise rotation of the text. The value must be 0, 90, 180, or 270.
VALIGN	The vertical alignment of the text. The value must be TOP, CENTER, or BOTTOM.
HALIGN	The horizontal alignment of the text. The value must be LEFT, CENTER, or RIGHT.
WORDWRAP	Boolean value. The text will be wrapped when the value is set to 1.
FONTCHARACTERSET	Used to display True Type fonts that are of different character sets. For example, Windings is **SYMBOL_CHARSET=2**. You must specify the FONTCHARACTERSET key in the xml file: `FONT=" Windings" FONTCHARACTERSET="2"`

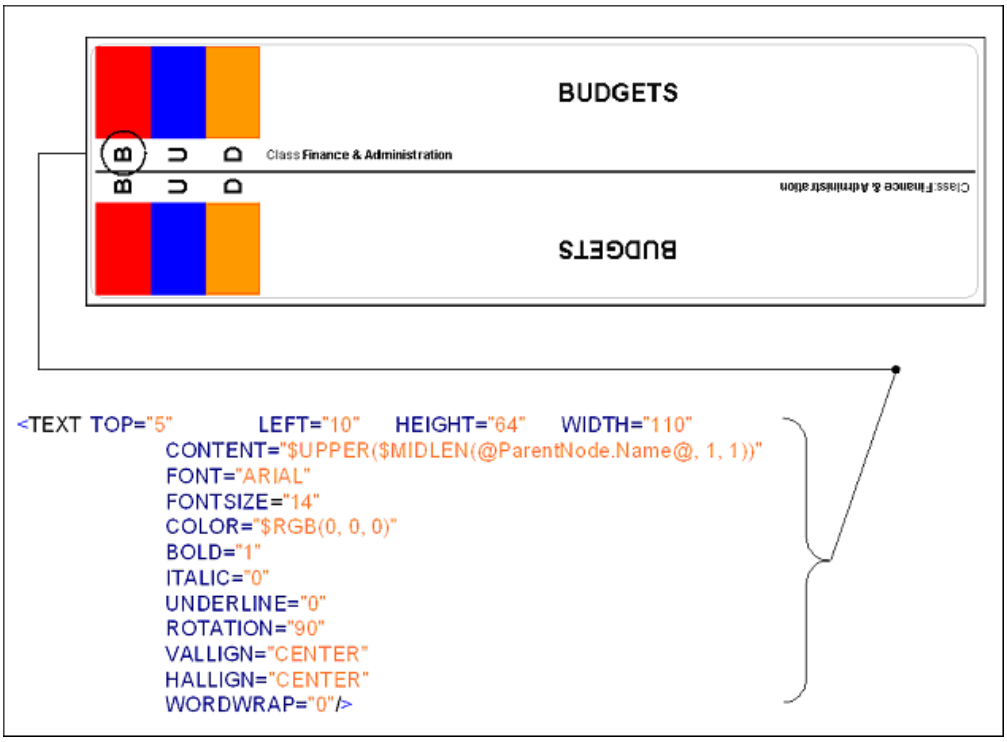


Figure 1-5: The <TEXT> element

Figure 1-6: The <TEXT> element

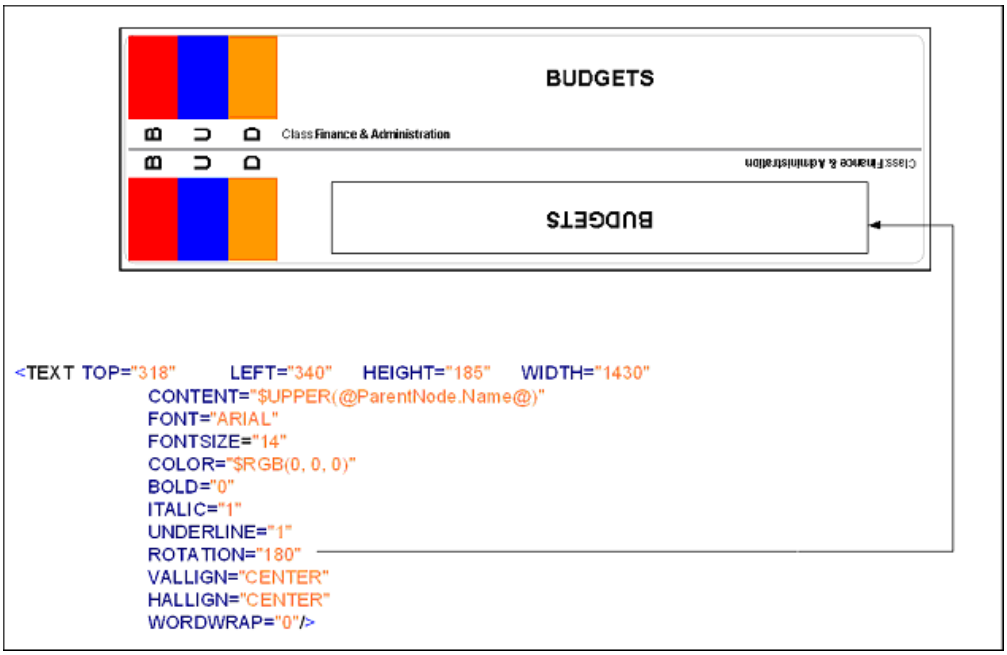


Figure 1-7: The <TEXT> element

<LINE>

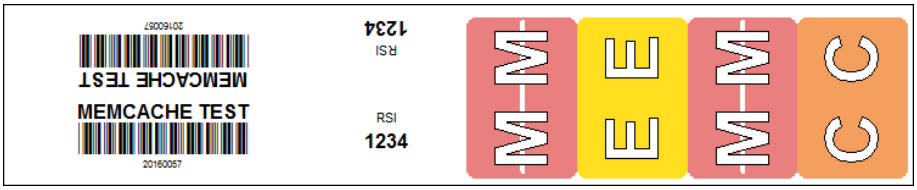
The <LINE> element is an optional element that specifies the location and attributes of a line on the label. This element contains the following attributes:

Table 1-8: LINE attributes

Attribute	Description
TOP	The measured distance from the top of the block definition to the top of the bounding box that will contain the line. The line will be drawn from the top left corner to the bottom right corner of this box. Measured in 1/10 of a millimeter
LEFT	The measured distance from the left of the block definition to the left of the bounding box that will contain the line. Measured in 1/10 of a millimeter
HEIGHT	The height of the bounding box that will contain the line. Measured in 1/10 of a millimeter
WIDTH	The width of the bounding box that will contain the line. Measured in 1/10 of a millimeter
COLOR	The color of the line
LINEWIDTH	The width of the line. Measured in 1/10 of a millimeter
TRANSPARENT	Places the object drawn with that attribute behind any other painted object in the same space.

Example 1-1: TRANSPARENT attribute

```
<LINE TOP="50" LEFT="1160" HEIGHT="325" WIDTH="0"
COLOR="$COLORMAP($TEXTMAP(SINGLEBAR,
$MIDLEN(@CurrentNode.Name@, 1, 1)),TRANSPARENT)"
LINEWIDTH="10" ROTATION="270"/>
```



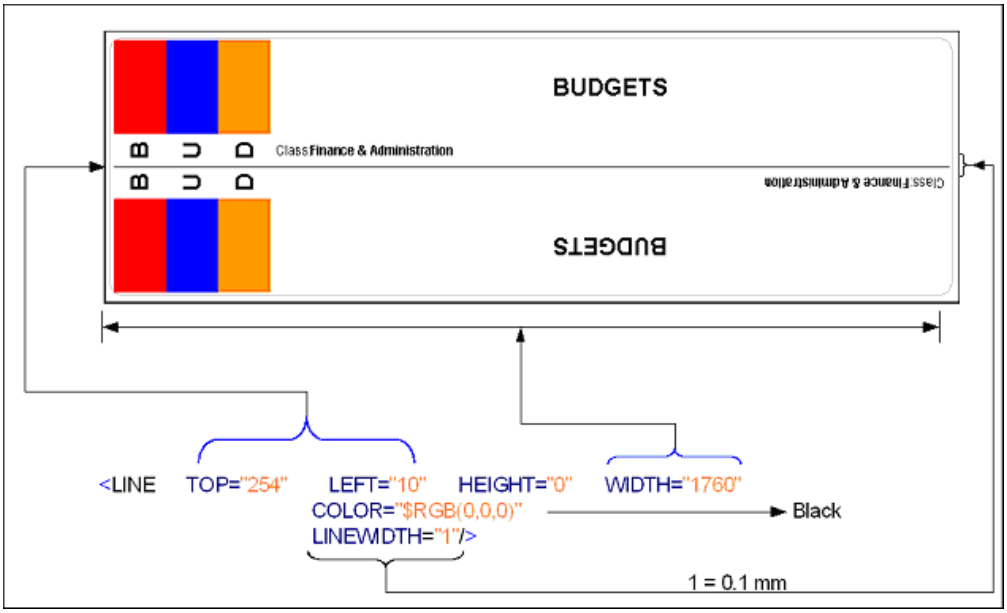


Figure 1-8: The <LINE> element

<RECTANGLE>

The <RECTANGLE> element is an optional element that specifies the location and attributes of a rectangle on the label. This element contains the following attributes:

Table 1-9: RECTANGLE attributes

Attribute	Description
TOP	The measured distance from the top of the block definition to the beginning of the rectangle. Measured in 1/10 of a millimeter
LEFT	The measured distance from the left of the block definition to the beginning of the rectangle. Measured in 1/10 of a millimeter
HEIGHT	The height of the rectangle. Measured in 1/10 of a millimeter
WIDTH	The width of the rectangle. Measured in 1/10 of a millimeter
FILLCOLOR	The fill color of the rectangle
BORDERCOLOR	The color of the border of the rectangle
BORDERWIDTH	The width of the border of the rectangle. Measured in 1/10 of a millimeter

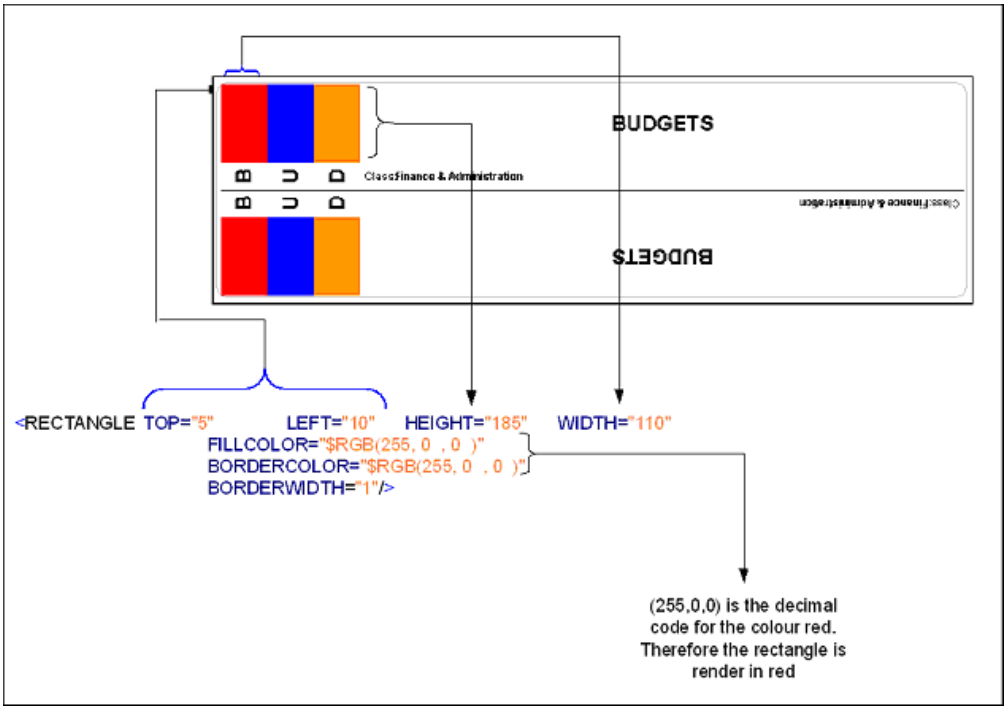


Figure 1-9: The <RECTANGLE> element

<ROUNDRECTANGLE>

The <ROUNDRECTANGLE> element is an optional element that specifies the location and attributes of a rectangle on the label. This element contains the following attributes:

Table 1-10: ROUNDRECTANGLE attributes

Attribute	Description
TOP	The measured distance from the top of the block definition to the beginning of the rectangle. Measured in 1/10 of a millimeter
LEFT	The measured distance from the left of the block definition to the beginning of the rectangle. Measured in 1/10 of a millimeter
HEIGHT	The height of the rectangle. Measured in 1/10 of a millimeter
WIDTH	The width of the rectangle. Measured in 1/10 of a millimeter
FILLCOLOR	The fill color of the rectangle
BORDERCOLOR	The color of the border of the rectangle
BORDERWIDTH	The width of the border of the rectangle. Measured in 1/10 of a millimeter
ELLIPSSEHEIGHT	The height of the corner of the rectangle. Measured in 1/10 of a millimeter
ELLIPSEWIDTH	The width of the corner of the rectangle. Measured in 1/10 of a millimeter

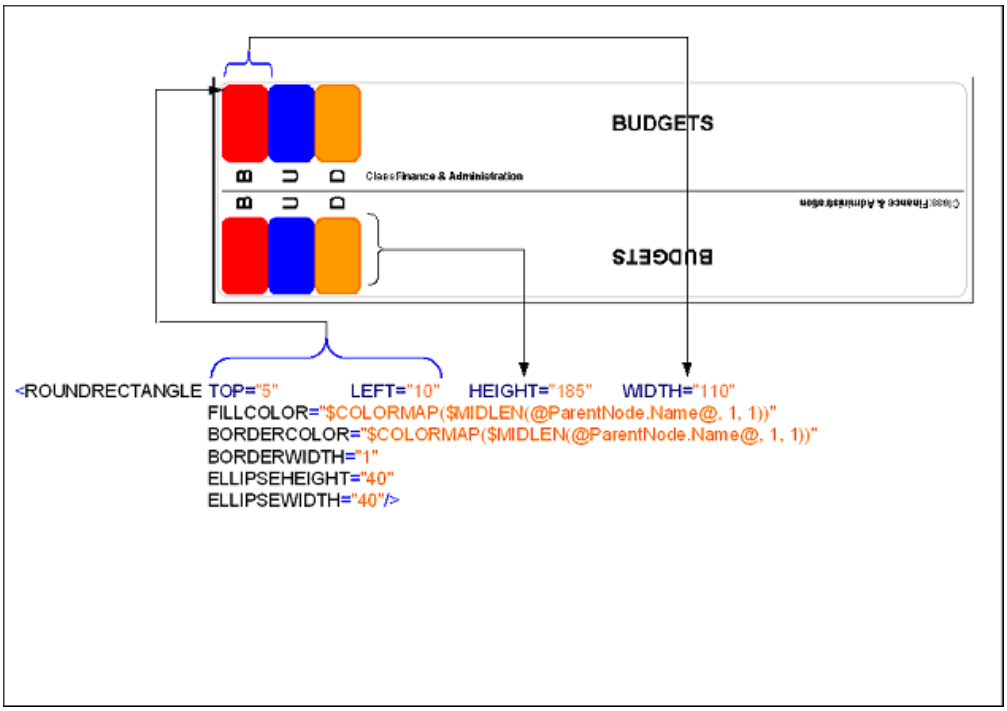


Figure 1-10: The <ROUNDRECTANGLE> element

<COLORMAP> and <COLOR>

The <COLORMAP> container element maps a text string to a specific color. Typically, this is used in conjunction with the <RECTANGLE> element so that each letter in the alphabet or number can be represented with a unique color. This element contains the following attribute:

Table 1-11: COLORMAP attribute

Attribute	Description
DEFAULTCOLOR	Represents the default color to use if a character is not mapped Must be in the form of $RGB(#, #, #), where # represents a numerical value in the Red/Green/Blue Color scale; for example, $RGB(255, 255, 255) For an RGB color scale chart, see "Appendix B" on page 35.

The <COLOR> element is a child element of <COLORMAP> and specifies what color to use when a specific character is mapped. This element contains the following attributes:

Table 1-12: COLOR attributes

Attribute	Description
TEXT	The text string being mapped. This text is case insensitive.
COLOR	Represents the color to use for the specified character Must be in the form of $RGB(#, #, #), where # represents a numerical value in the Red/Green/Blue Color scale; for example, $RGB(255, 255, 255) For an RGB color scale chart, see "Appendix B" on page 35.

<TEXTMAP> and <TEXTMAPITEM>

The <TEXTMAP> container element maps a string to another string. The element is used in conjunction with the $TEXTMAP string function. This element contains the following attribute:

Table 1-13: TEXTMAP attribute

Attribute	Description
NAME	A unique identifier representing a set of text mappings

The <TEXTMAPITEM> element is a child element of <TEXTMAP> and specifies the mapping relationship between the strings. This element contains the following attributes.

Table 1-14: TEXTMAPITEM attributes

Attribute	Description
TEXT	The string value that is being mapped
VALUE	The string that the value of TEXT is being mapped to

1.2 Label Content

The following is a list of database fields, tables, and string functions that can be used to control the output of text on the label.

1.2.1 Database Content for Physical Items

The following is a list of database fields that can be accessed and displayed in a label for a physical item. To display the content of a database field, you must delimit the field with the @ symbol. For example, to display the name of the physical item on a label, insert the following attribute in a <TEXT> element:

```
CONTENT="@CurrentNode.Name@"
```

CurrentNode

The `CurrentNode` object returns data about the currently selected physical item. Any column from the Dtree table that can have its value displayed directly can be accessed. The following is a list of the available columns.

Note: The ExtendedData column cannot be accessed.

Table 1-15: Column available for the CurrentNode object

Column name	Description
Name	The name of the physical object, which is also the content of the **Name** field in the **General** tab for the physical object

ParentNode

The `ParentNode` object returns data about the parent of the currently selected physical item. Any column from the Dtree table that can have its value displayed directly can be accessed. The following is a list of the available columns.

Note: The ExtendedData column cannot be accessed.

Table 1-16: Column available for the ParentNode object

Column name	Description
Name	The name of the object that contains the physical item, which is also the content of the **Name** field in the **General** tab for the physical object's parent

PhysItemCO

The `PhysItemCO` object returns charge-out information pertaining to the physical object. The following is a list of the available columns.

Table 1-17: Columns available for PhysItemCO

Column name	Description
CopyNumber	The number assigned to the copy
Location	The location of the physical record
UniqueID	The unique identifier for the physical object
ChargeTo	The user log-in or external client name representing the user who charged out the physical object
ObtainedBy	The user who receives the physical item on behalf of the borrower.
UserID	The unique identifier representing the user who charged out the physical object
ObtainedByID	The unique identifier representing the user who received the physical item on behalf of the borrower.
BorrowPerformerID	The unique identifier representing the user who recorded the loan.
ChargeDate	The date the physical object was charged out
ActualBorrowDate	Date the physical item was actually borrowed.
ReturnBy	The data the physical object should be returned by
BorrowComment	The comment created when the physical object was borrowed

PhysItemData

The `PhysItemData` object returns information pertaining to the physical object. The following is a list of the available columns.

Table 1-18: Columns available for PhysItemData

Column name	Description
MediaTypeID	A numerical representation of the media type For example, 3735 = Physical Folder 3736 = Physical Document
DefaultLocation	The location of the physical record
FromDate	The date used to identify the starting date of the contents of a physical item

Column name	Description
ToDate	The date used to identify the ending date of the contents of a physical item
ContainerNumber	The number identifying container type for physical items within it
PhysItemKeywords	Text that is indexed against the physical item

PhysItemExt

The `PhysItemExt` object returns information pertaining to the list of names of media type extension fields (not display names) for a specific physical item.

Table 1-19: Column available for PhysItemExt

Column name	Description
XXXX	Extension fields by name

ClassLevel

The `ClassLevel` object returns information pertaining to the RM Classification data for a specific physical item.

Table 1-20: Columns available for ClassLevel

Column name	Description
FileNumber	File number of physical item's classification
Name	Returns the full name of physical item's classification
Name:-2	Returns the second name from the bottom of the RM Classification. The -2 is configurable. It is used to determine the number (2) above the bottom classification level.
Name:2	Returns the second name from the top RM Classification. The 2 is configurable. It is used to determine the number (2) below the top classification level.
Current	Returns which level (number) is the current classification

LocationLevel

The `LocationLevel` object returns information pertaining to the Content Server location data for a specific physical item.

Table 1-21: Columns available for LocationLevel

Column name	Description
Name	Returns the full name of physical item's location
Name:-2	Returns the second name from the bottom of the Content Server location. The -2 is configurable. It is used to determine the number (2) above the bottom location level.
Name:2	Returns the second name from the top Content Server location. The 2 is configurable. It is used to determine the number (2) below the top location level.

Classification

The `Classification` object returns information pertaining to the RM Classification data for a specific physical item. It lists any of the columns from the RM_CLASSIFICATION table.

Table 1-22: Columns available for Classification

Column name	Description
RSI	Returns the RSI value of physical item's classification
FileNumber	File number assigned to the RM Classification code
FILE_STATUS	Corresponds to FILE_STATUS lookup code in FILE_STATUS table
Essential	Corresponds to ESS_RECORDS lookup code in ESS_RECORDS table
STORAGE	Corresponds to STORAGE lookup code in STORAGE table
Keywords	A string of keywords for the RM Classification
Subject	A string with the subject of the RM Classification
DispAuthority	Disposition Authority

ObjectClassification

The `ObjectClassification` object returns information pertaining to the RM Classification data for a specific physical item. It lists any of the columns from the rimsNodeClassification table.

Table 1-23: Columns available for ObjectClassification

Column name	Description
rimsAccession	Corresponds to ACCESSION in ACCESSION table
rimsAddressee	Corresponds to the **Records Detail Addressee** field
rimsSubject	Corresponds to the **Records Detail Subject** field
rimsDocDate	Returns the Record Date value of physical item's classification
rimsEssential	Corresponds to the **Records Detail Essential** field
rimsEstablishment	Corresponds to the **Records Detail Originating Organization**
rimsOfficial	Corresponds to the **Records Detail Official** check box
rimsOriginator	Corresponds to the **Records Detail Author** or **Originator** field
rimsRSI	Corresponds to the **Records Detail RSI field**. Inherited from RM_CLASSIFICATION table upon classification.
rimsSentTo	Corresponds to the **Detail Other Addressee Records**
rimsStatus	Returns the Record Status value of physical item's classification
rimsStatusDate	Corresponds to the **Records Detail Status Date** field. Inherited from RM_CLASSIFICATION table upon classification

ParentPhysItemCO

The `ParentPhysItemCO` object returns information pertaining to the circulation data for the parent of the physical item. It lists any of the columns from the PhysItemCO table. For information about the PhysItemCO table, see "PhysItemCO" on page 19.

PhysObjType

The `PhysObjType` object returns information pertaining to the Physical Object type data for a specific physical item.

Table 1-24: Column available for PhysObjType

Column name	Description
Name	Returns the Physical Object Type Name value of the item

Boxinfo

If a physical item is assigned to a box, the `Boxinfo` object returns all metadata about that box. It lists columns from the DTree and PhysItemCO tables. All fields from the DTree table can be accessed. For information about the PhysItemCO table, see "PhysItemCO" on page 19.

Categories

The `Categories` object returns information pertaining to the Cats and Atts data for a specific physical item. It lists categories and attributes with the format Category Name: Attribute Name.

Label Design supports the Categories and Attributes Extensions and the Automatic Document Number modules. Attributes for these modules are supported fields that can be accessed and displayed in a label for a physical item.

1.2.2 Custom Tables

You can choose attributes from a customized table and display them in labels for physical items. Data stored in custom tables is accessed differently than data that is stored in tables known to Physical Objects module. To retrieve data from a custom table, a user must provide a *where* clause, which is the name of the custom table and a list of columns to select.

Example 1-2: Entry in Labels Script to Retrieve CollectionID from Collections Table

Insert the following values into the **LabelDefSpecs** columns:

For **DefSpecs**, insert: `CustomTable1`

For **LabelDefName**, insert: `N'SAMPLE 1`

For **ColumnsToSelect**, insert: `CollectionID`

For **TableName**, insert: `Collections Where DataID`

In the label definition, insert: `CONTENT="@CustomTable1.CollectionID@"`

Displaying Multilingual Names and Descriptions in Labels

You can display the name and description in a different language by modifying the LabelDefSpecs columns.

Example 1-3: Display German Name and Description

Insert the following values into the **LabelDefSpecs** columns:

For **TableName**, insert `WebNodesMeta_de WHERE DataID`

This specifies that the data will be in German.

1.2.3 Database Content for Locators

The following are the database fields that can be accessed and displayed in a label for a locator. To display the content of a database field, you must delimit the field with the @ symbol.

StorageLocator

The StorageLocator object returns information pertaining to the locator data. It lists any of the columns from the storage locator.

Table 1-25: Columns available for StorageLocator

Column name	Description
Facility	Name of the storage facility code. Corresponds to Facility code in the Facility table
Area	Name of the storage area code. Corresponds to Area code in the Area table
BoxLocator	Name of the storage locator. Locator can be created only for valid Facility and Area codes
LocatorType	Locator type code. Corresponds to Locator Type in the LocatorType table
RefRate	Reference rate code. Corresponds to RefRate code in the RefRate table
TotalSpace	Total space of a locator
FreeSpace	Free space of a locator
StorLocDepth	Depth of a locator. For a facility managed by depth, the value is checked when the locator is assigned.
StorLocHeight	Height of a locator. For a facility managed by height, the value is checked when the locator is assigned.
StorLocLength	Length of a locator. For a facility managed by length, the value is checked when the locator is assigned.
LocatorSortKey	System-generated value that enables proper sorting of alphanumeric values in the **BoxLocator** field. Zero filled to six characters.

1.2.4 Database Content for Users

The following are the database fields that can be accessed and displayed in a label for a Content Server user. To display the content of a database field, you must delimit the field with the @ symbol.

Note: Unlike XML tags and attributes, the database tables and field names are not case sensitive.

MainUserInfo

User info is derived from the KUAF table in Content Server, any value can be used to generate a user label. The following table provides a sample of some of the attributes available from the KUAF table.

Table 1-26: MainUserInfo Columns – sample of KUAF table values

Column Name	Description
ID	Content Server User ID
Name	Sign in name of the Content Server user
GroupID	Content Server Group ID
LastName	The last name of the Content Server user
MiddleName	The middle name of the Content Server user
FirstName	The first name of the Content Server user
MailAddress	The mailing address of the Content Server user

Department

Table 1-27: Department Columns

Column Name	Description
Name	The name of the user's main department

AllUserGroups

Table 1-28: AllUserGroups Columns

Column Name	Description
Name	When the AllUserGroups key is added in a label definition, the names of all groups to which the user belongs are sent to the label. To display the name of an additional group, add the group name between the @ symbol. For example: CONTENT="@Group2@" CONTENT="Group3"

1.2.5 String Functions

The following is a list of string functions that can be performed on the text found in the `CONTENT` attribute of the `<TEXT>` element.

UPPER Function

Returns a string that has been converted to uppercase

$UPPER(string)

Arguments: *String* - The string argument is any valid string expression.

Example:

`$UPPER(Hello World)` will return `HELLO WORLD`.

LOWER Function

Returns a string that has been converted to lowercase

$LOWER(string)

Arguments: String - The string argument is any valid string expression.

Example:

`$LOWER(Hello World)` will return `hello world`.

MIDLEN Function

Returns a specified number of characters from a string

$MIDLEN(string, StartPos, NumChars)

Arguments: String - The string argument is any valid string expression.

StartPos - Character position in string at which the part to be taken begins (the first character has an index of '1').

NumChars - Number of characters to return. If there are fewer than NumChars available, all the characters from the start position to the end of the string are returned.

Example:

`$MIDLEN(Wow XML is fun!, 5, 3)` will return `XML.`

MID Function

Returns the end portion of a string based on the specified starting position

$MID(string, StartPos)

Arguments: String - The string argument is any valid string expression.

StartPos - Numeric expression indicating the starting position of the string to return. The index of the first character is 1. If greater than the number of characters in the string, a zero-length string("") is returned.

Example:

`$MID(Wow XML is fun!, 5)` will return `XML is fun!`

LEFT Function

Returns a specified number of characters from the left side of a string

$LEFT(string, NumChars)

Arguments: String - The string argument is any valid string expression.

NumChars - Numeric expression indicating how many characters to return. If 0, a zero-length string("") is returned. If greater than or equal to the number of characters in the string, the entire string is returned.

Example:

`$LEFT(Wow XML is fun!, 3)` will return `Wow.`

RIGHT Function

Returns a specified number of characters from the right side of a string

$RIGHT(string, NumChars)

Arguments: String - The string argument is any valid string expression.

NumChars - Numeric expression indicating how many characters to return. If 0, a zero-length string is returned. If greater than or equal to the number of characters in the string, the entire string is returned.

Example:

`$RIGHT(Wow XML is fun!, 4)` will return `fun!`

TRIM Function

Returns a copy of a string with leading and trailing spaces removed

$TRIM(string)

Arguments: String - The string argument is any valid string expression.

Example:

`$TRIM(LLRM is Great! )` will return `LLRM is Great!`

TRIMLEFT Function

Returns a copy of a string without the leading spaces

$TRIMLEFT(string)

Arguments: String - The string argument is any valid string expression.

Example:

`$TRIMLEFT( Hello!)` will return `Hello!`

TRIMRIGHT Function

Returns a copy of a string without the trailing spaces

$TRIMRIGHT(string)

Arguments: String - The string argument is any valid string expression.

Example:

`$TRIMRIGHT(Hello! )` will return `Hello!`

CHR Function

Returns the character associated with the specified character code

$CHR(charcode)

Arguments: Charcode - The charcode argument is a number that identifies a character.

Example:

`$CHR(37)` will return `%`.

`$CHR(62)` will return `>`.

IFEQUAL Function

Tests two strings for equality and returns a string based on the results

$IFEQUAL(string1, string2, EqualText, NotEqualText)

Arguments: string1 - The string1 argument is any valid string expression.

string2 - The string2 argument is any valid string expression.

EqualText - A string value to return if string1 and string2 are identical

NotEqualText - The string value to return if string1 and string2 are not identical

Example:

`$IFEQUAL(XML, XML, Matched!, Not Matched!)` will return `Matched!`

`$IFEQUAL(XML, HTML, Matched!, Not Matched!)` will return `Not Matched!`

TEXTMAP Function

Maps a string to another string using a provided lookup table

$TEXTMAP(TextMapName, string)

Arguments: TextMapName - The name of the <TEXTMAP> to use (see <TEXTMAP> under the Label Definitions section)

String - The string argument is any valid string expression.

Example:

<TEXTMAP NAME="abbrev">
<TEXTMAPITEM TEXT="OTC" VALUE="OpenText Corporation"/>
<TEXTMAPITEM TEXT="LL" VALUE="contentserver"/>
<TEXTMAPITEM TEXT="RM" VALUE="Records Management"/>

</TEXTMAP>

`$TEXTMAP(abbrev, CS)` will return `Content Server.`

`$TEXTMAP(abbrev, RM)` will return `Records Management.`

1.3 The Label Script File

Up to this point in the documentation, the XML for the label stock definitions and the label definitions have been referenced as separate XML documents. For XML-validation purposes, it is beneficial to store them as separate XML instances; however, in reality, to insert the XML data into Content Server, they exist in one file that is not a true XML instance.

The input file that contains the XML for both the label stock definitions and label definitions is called Labels.wts. The label.wts file is an SQL script file and can be found in the \module\physicalobjects_x_x_x directory of the Content Server installation directory. The name and location of the file is hardcoded and cannot change.

The label script file consists of SQL insert and delete statements. The delete statements ensure that the script file can be re-run without the risk of duplicating the XML data.

Note: The intent of these delete statements is so that names given to new label stock or label definitions cannot conflict with existing names. If a conflict occurs, they are deleted from Content Server.

The script must contain a section for each of the following three tables: LabelStock, LabelDefinition, and LabelDefSpecs. The LabelStock section contains data defining the label stock. The LabelDefinition section contains data defining the content and format of the data that will be displayed on the label. The LabelDefSpecs section contains the data defining how the information is selected from the database.

The Label.wts file has the following generic structure:

```
delete from LabelStock where StockName = '$SNAME';
insert into LabelStock (StockName, LabelsPerPage, StockDefXML) values
('$SNAME', 1 ,:1)
\
$LONG
    <INSERT LABELSTOCKDEFINITION HERE>
//
;



delete from LabelDefinition where LabelDefName = 'SAMPLE 1';
insert into LabelDefinition (LabelDefName, StockName,LabelType,
LabelDefXML) values ('SAMPLE 1', '7X2LABEL' ,0 , :1)
\
$LONG
    <INSERT LABELDEFINITION HERE>
//
;


delete from LabelDefSpecs where LabelDefName = 'SAMPLE 1'
insert into LabelDefSpecs (DefSpecs, LabelDefName, ColumnsToSelect)
values ('ParentNode', 'SAMPLE 1',:1)
\
$LONG
Name
//
;
```

Figure 1-11: Sample Label.wts script file

Note: LabelType = '0' is for physical items and LabelType = '1' is for locators.

Chapter 2

Appendix A

Measurement Conversion Table

Table 2-1: Measurement conversion table

Inches	1/10 of mm		Inches	1/10 of mm
1	254		7	1778
1.25	317.5		7.25	1841.5
1.5	381		7.5	1905
1.75	444.5		7.75	1968.5
2	508		8	2032
2.25	571.5		8.25	2095.5
2.5	635		8.5	2159
2.75	698.5		8.75	2222.5
3	762		9	2286
3.25	825.5		9.25	2349.5
3.5	889		9.5	2413
3.75	952.5		9.75	2476.5
4	1016		10	2540
4.25	1079.5		10.25	2603.5
4.5	1143		10.5	2667
4.75	1206.5		10.75	2730.5
5	1270		11	2794
5.25	1333.5		11.25	2857.5
5.5	1397		11.5	2921
5.75	1460.5			
6	1524			
6.25	1587.5			
6.5	1651			
6.75	1714.5			

Chapter 3

Appendix B

RBG Color Scale

Table 3-1: RGB color scale

RGB Decimal Representation	Color
$RGB(255, 255, 255)	
$RGB(204, 204, 204)	
$RGB(153, 153, 153)	
$RGB(102, 102, 102)	
$RGB(51, 51, 51)	
$RGB(0, 0, 0)	
$RGB(64, 0, 0)	
$RGB(128, 0, 0)	

RGB Decimal Representation	Color
$RGB(128, 64, 0)	
$RGB(128, 128, 0)	
$RGB(192, 128, 0)	
$RGB(255, 0, 0)	
$RGB(255, 64, 0)	
$RGB(255, 128, 0)	
$RGB(255, 192, 0)	
$RGB(255, 255, 0)	
$RGB(255, 255, 128)	
$RGB(128, 255, 64)	
$RGB(64, 255, 128)	

RGB Decimal Representation	Color
$RGB(0, 255, 0)	
$RGB(64, 192, 0)	
$RGB(0, 128, 64)	
$RGB(0, 64, 64)	
$RGB(0, 64, 128)	
$RGB(0, 128, 128)	
$RGB(0, 0, 64)	
$RGB(0, 0, 160)	
$RGB(0, 0, 255)	
$RGB(0, 128, 192)	
$RGB(128, 255, 255)	

RGB Decimal Representation	Color
$RGB(128, 0, 192)	
$RGB(128, 64, 255)	
$RGB(192, 0, 255)	
$RGB(255, 0, 192)	
$RGB(255, 128, 192)	