HP Exstream

Version 6.1 - Version 7.0

Course 210:
Dialogue System Administration

STUDENT GUIDE

i n v e n t

210: Advanced data concepts

Contents

Chapter 1: Class Overview

1.1 Advanced Data Concepts

This three-day course provides details and hands-on experience creating data structures that use different types of data files, complex rules and formulas, variables, and inputs from XML files.

Audience

Individuals responsible for creating and maintaining the data structures needed for Dialogue applications.

Prerequisites

101 Introduction to Dialogue, 110 Dialogue System Administration, experience working with data structures.

Goals

To create effective HP Dialogue data structures you should be able to:

- Add variables and data file descriptions to the design database using Design Manager
- Create data maps that link variables with data files
- Create data structures that:
  - Expedite variable and data map creation by using COBOL Copybooks
  - Add business rules and formulas to the design database
  - Extract variable information from an existing print file using the Print Miner module
  - Access relational databases using the ODBC Access module
  - Include external images and text files by using the Dynamic content import module
  - Incorporate data into Dialogue using the XML Input module
  - Connect to any corporate database by using the Dynamic Data Access module

1.2 Logistics

How to log on to Dialogue

Username: admin

Password: xxx (lower case)

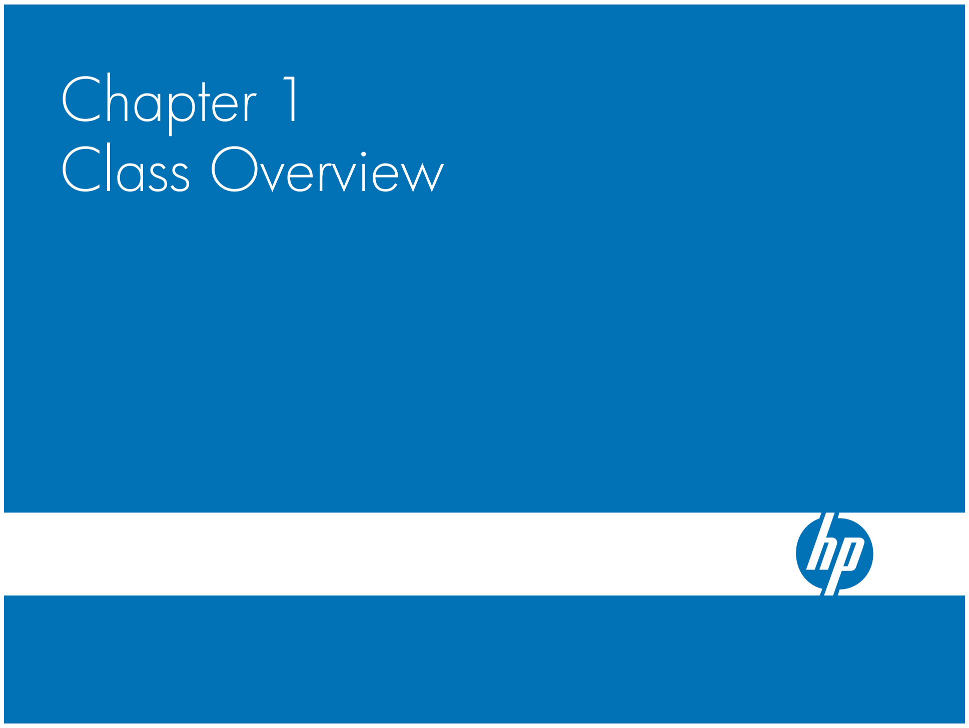
Location of Working Files

You will find all files used in training in the following directory:

C:\210 Advanced Data Concepts\

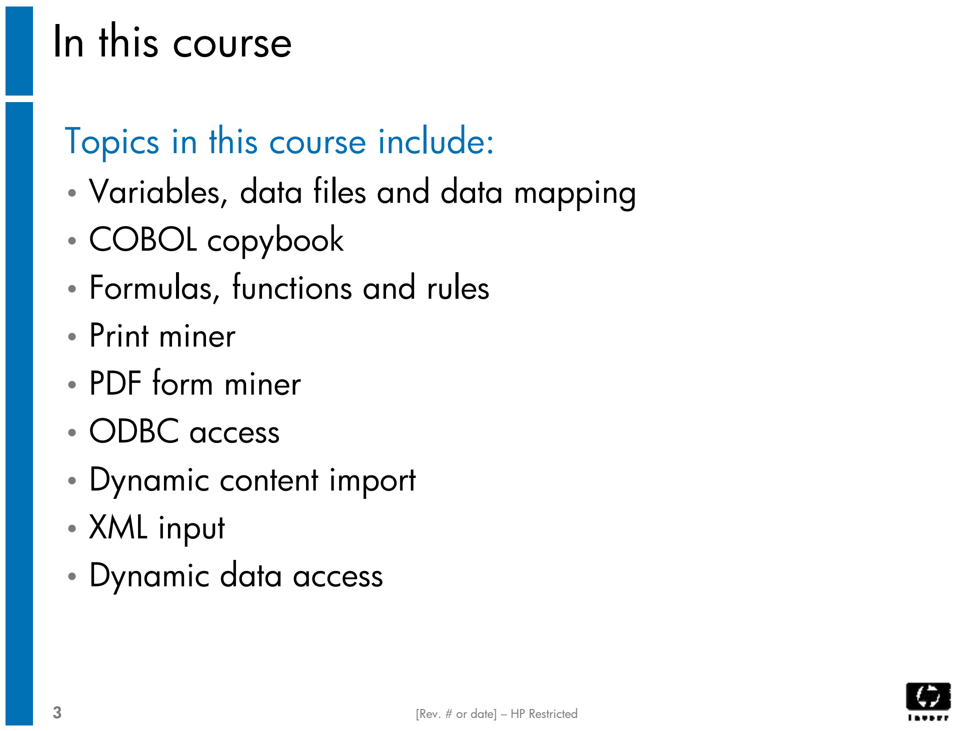
- Class DB
- Data Files
- Graphic Files

1.3 In this Course



Chapter 1
Class Overview

In this Course



In this course

Topics in this course include:

- Variables, data files and data mapping
- COBOL copybook
- Formulas, functions and rules
- Print miner
- PDF form miner
- ODBC access
- Dynamic content import
- XML input
- Dynamic data access

3 [Rev. # or date] – HP Restricted

For more information

For more information, refer to the following Dialogue documentation, listed in alphabetical order.

- Data Files guide DataFile.pdf
- Dynamic Data Access guide DDA.pdf
- Dynamically Importing Content guide Dynamic.pdf
- Rules, Formulas, and Functions guide Formulas.pdf
- Logic Designer guide Logic.pdf
- ODBC Access guide ODBC.pdf
- Populating and Mining PDF Forms guide PDFForm.pdf
- Print Miner guide Miner.pdf
- System Administration guide System.pdf
- Table Processing guide Tables.pdf
- Variables guide Variable.pdf
- XML Input guide XMLIn.pdf

Chapter 2: Variables

2.1 In this Chapter

Variables are values that can change, depending on conditions or information passed to the Dialogue Engine. They are basic to Dialogue's ability to build fully personalized documents.

In this chapter you learn to add variables and their descriptions to the design database using Design Manager.

Objectives

By the end of this chapter, you should be able to:

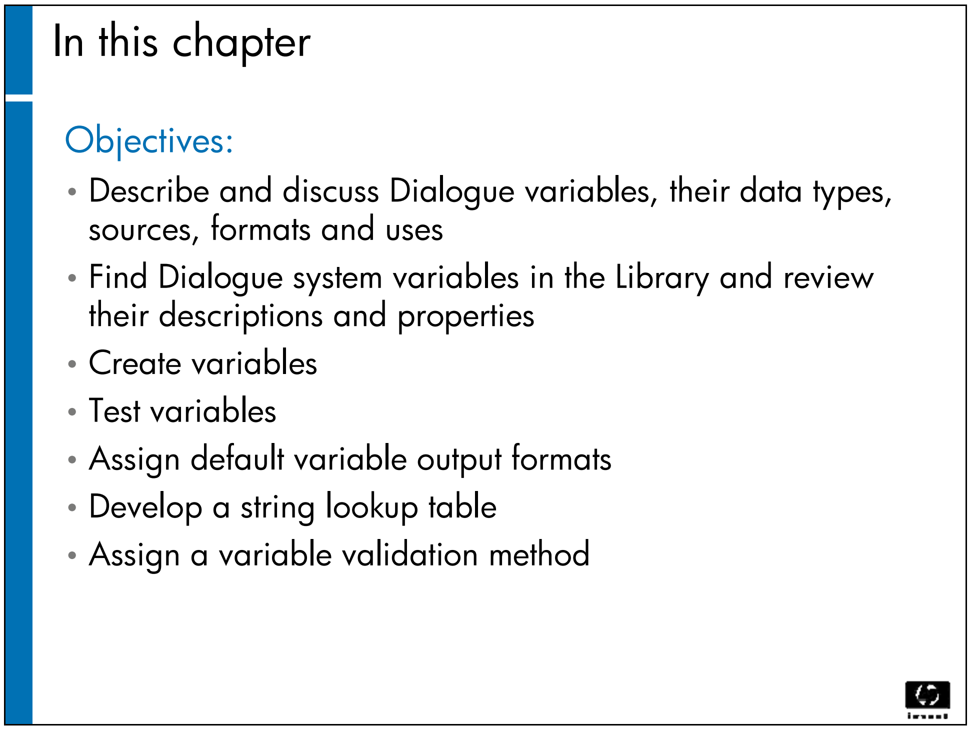
- Describe and discuss Dialogue variables, their data types, sources, formats, and uses.
- Find Dialogue system variables in the Library and review their descriptions and properties.
- Create variables.
- Test variables.
- Assign default variable output formats.
- Develop a string lookup table.
- Assign a variable validation method.

For more information

For more information, refer to the following Dialogue documentation.

Variables guide	Variable.pdf
System Administration guide	System.pdf

In this Chapter

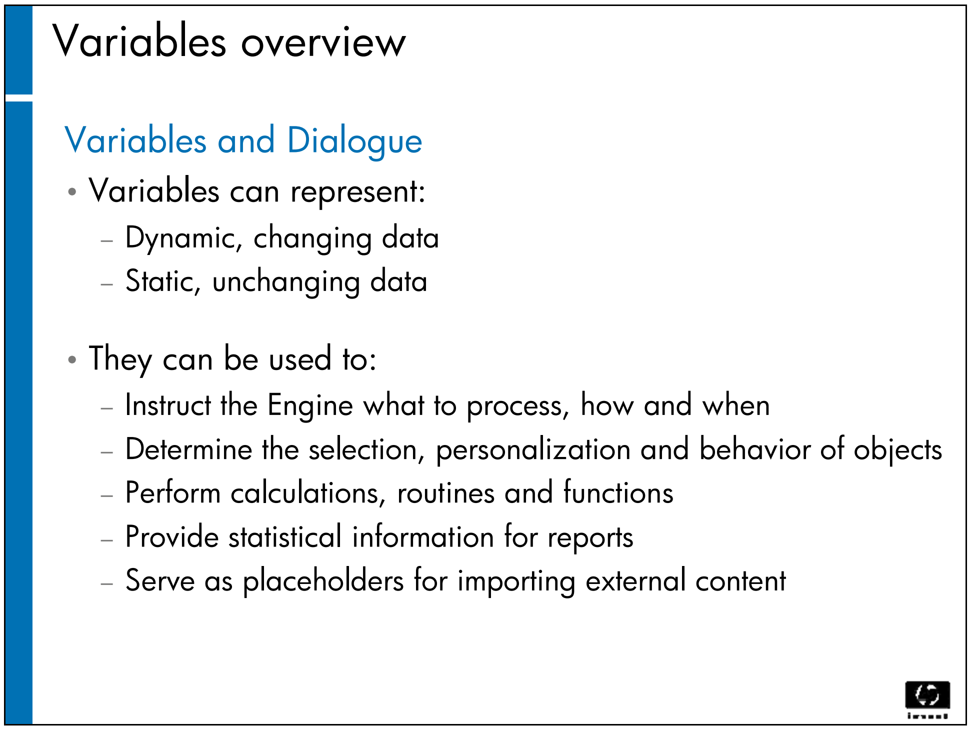


In this chapter

Objectives:

- Describe and discuss Dialogue variables, their data types, sources, formats and uses
- Find Dialogue system variables in the Library and review their descriptions and properties
- Create variables
- Test variables
- Assign default variable output formats
- Develop a string lookup table
- Assign a variable validation method

2.2 Variables Overview

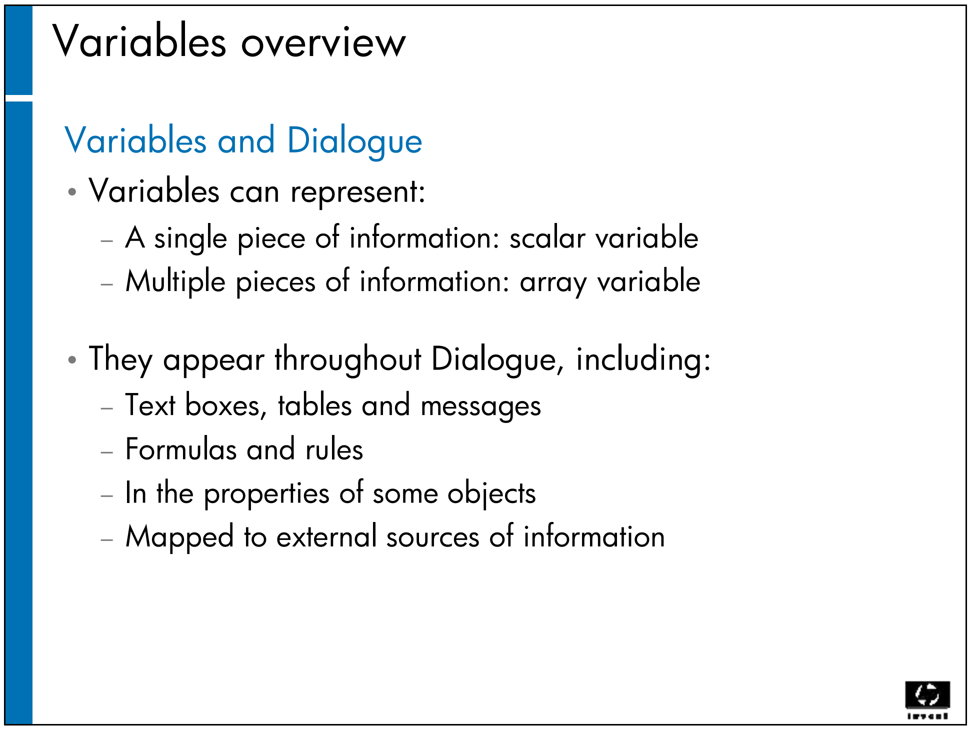


Variables overview

Variables and Dialogue

- Variables can represent:
  - Dynamic, changing data
  - Static, unchanging data
- They can be used to:
  - Instruct the Engine what to process, how and when
  - Determine the selection, personalization and behavior of objects
  - Perform calculations, routines and functions
  - Provide statistical information for reports
  - Serve as placeholders for importing external content

The capabilities of variables expand with each Dialogue software release.



Variables overview

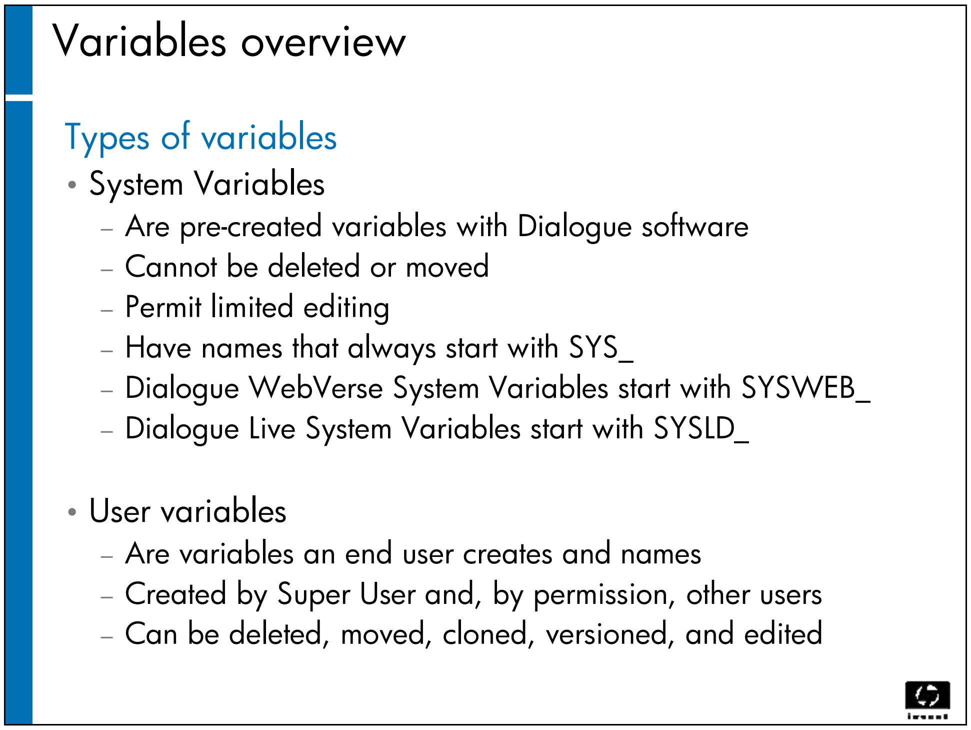
Variables and Dialogue

- Variables can represent:
  - A single piece of information: scalar variable
  - Multiple pieces of information: array variable
- They appear throughout Dialogue, including:
  - Text boxes, tables and messages
  - Formulas and rules
  - In the properties of some objects
  - Mapped to external sources of information

A variable can be:

- As simple as a scalar variable representing a Boolean value (a yes or no).
- As complex as a large array representing many financial transactions.

Variables Overview

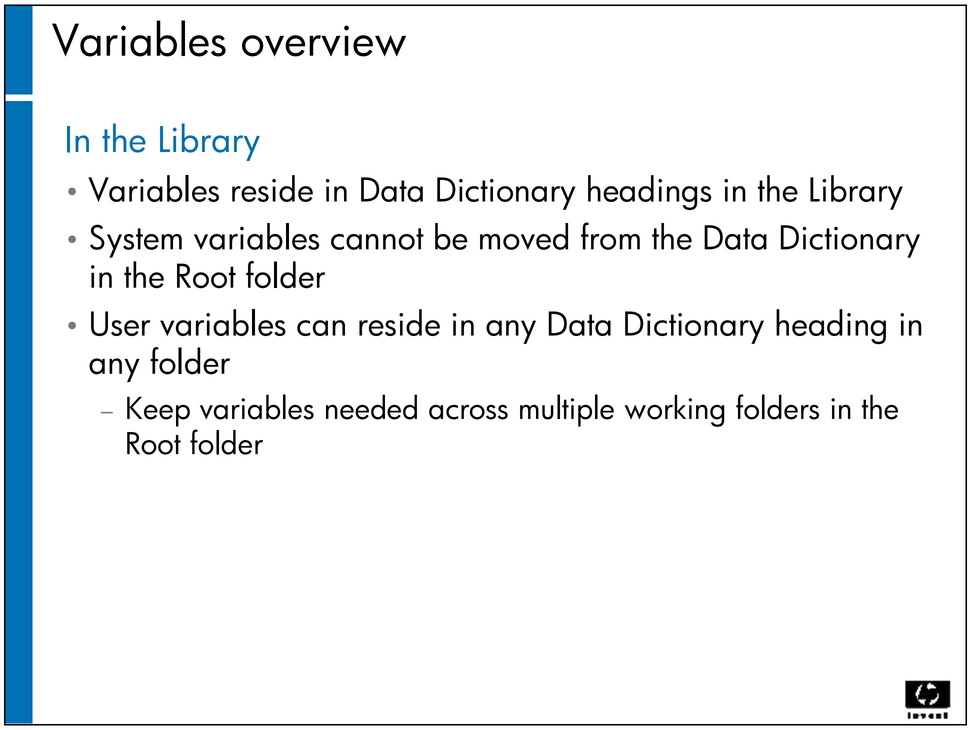


Variables overview

Types of variables

- System Variables
  - Are pre-created variables with Dialogue software
  - Cannot be deleted or moved
  - Permit limited editing
  - Have names that always start with SYS_
  - Dialogue WebVerse System Variables start with SYSWEB_
  - Dialogue Live System Variables start with SYSLD_
- User variables
  - Are variables an end user creates and names
  - Created by Super User and, by permission, other users
  - Can be deleted, moved, cloned, versioned, and edited

System variables are used to get current dates, number of documents processed, and even to record the paper type and amount of the paper stock being used. Some System Variables allow you to change property settings.



Variables overview

In the Library

- Variables reside in Data Dictionary headings in the Library
- System variables cannot be moved from the Data Dictionary in the Root folder
- User variables can reside in any Data Dictionary heading in any folder
  - Keep variables needed across multiple working folders in the Root folder

If you try to move a user variable into a folder that already has a variable with the same name, you will get a warning message. This prevents inadvertently writing over the properties of existing variables.

Position of Variable in a Library List

The position of a variable in a list has no effect on variable processing. Dialogue always arranges variables alphabetically to make them easy to find in the Library. Changing variable names to arrange them in ascending processing order has no effect.

Variable processing *can* be controlled by the order in which they appear in a formula.

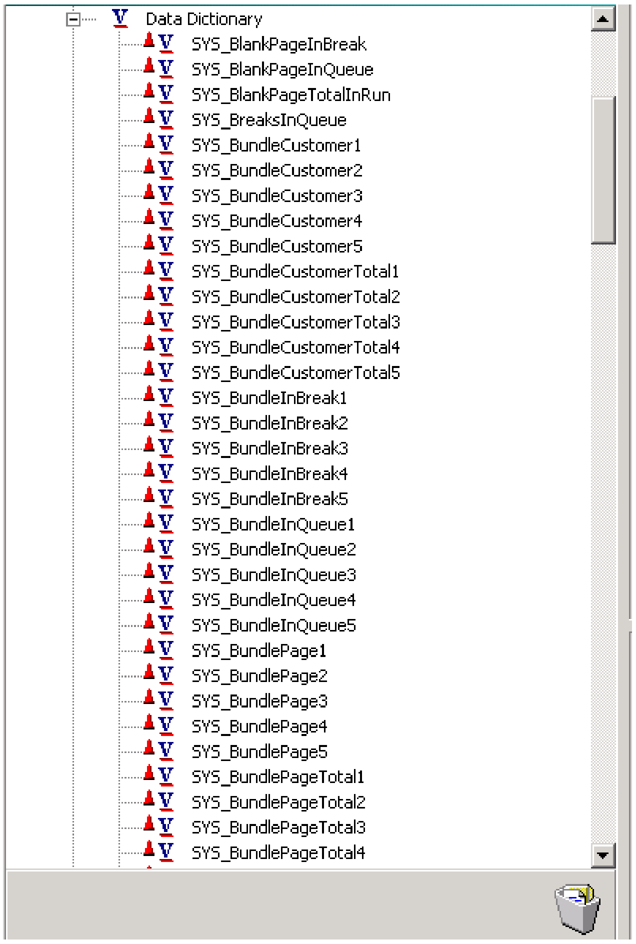
Variable Creation Date

The creation date of the variable may become the deciding factor of one variable processing time coming before another when *all* other factors are equal. Be aware that Load/Unload operations may change the creation date property of a variable.

Viewing the Data Dictionary

The variables in the **Data Dictionary** can be viewed in the Library by expanding a **Data Dictionary** heading.

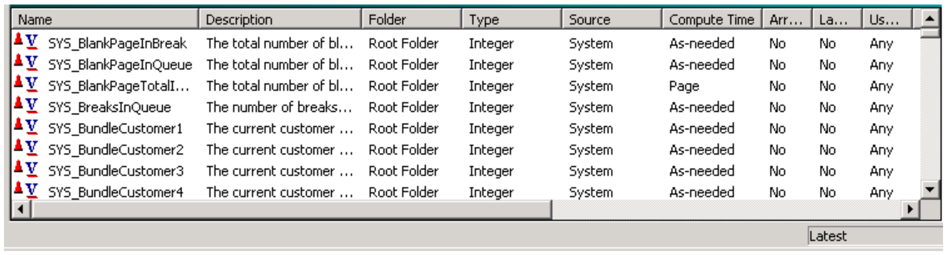
View Data Dictionary in the Library



```
Data Dictionary
    SYS_BlankPageInBreak
    SYS_BlankPageInQueue
    SYS_BlankPageTotalInRun
    SYS_BreaksInQueue
    SYS_BundleCustomer1
    SYS_BundleCustomer2
    SYS_BundleCustomer3
    SYS_BundleCustomer4
    SYS_BundleCustomer5
    SYS_BundleCustomerTotal1
    SYS_BundleCustomerTotal2
    SYS_BundleCustomerTotal3
    SYS_BundleCustomerTotal4
    SYS_BundleCustomerTotal5
    SYS_BundleInBreak1
    SYS_BundleInBreak2
    SYS_BundleInBreak3
    SYS_BundleInBreak4
    SYS_BundleInBreak5
    SYS_BundleInQueue1
    SYS_BundleInQueue2
    SYS_BundleInQueue3
    SYS_BundleInQueue4
    SYS_BundleInQueue5
    SYS_BundlePage1
    SYS_BundlePage2
    SYS_BundlePage3
    SYS_BundlePage4
    SYS_BundlePage5
    SYS_BundlePageTotal1
    SYS_BundlePageTotal2
    SYS_BundlePageTotal3
    SYS_BundlePageTotal4
```

Alternately, you can view details about the variables by dragging the **Data Dictionary** heading to the Property Panel or Edit Panel.

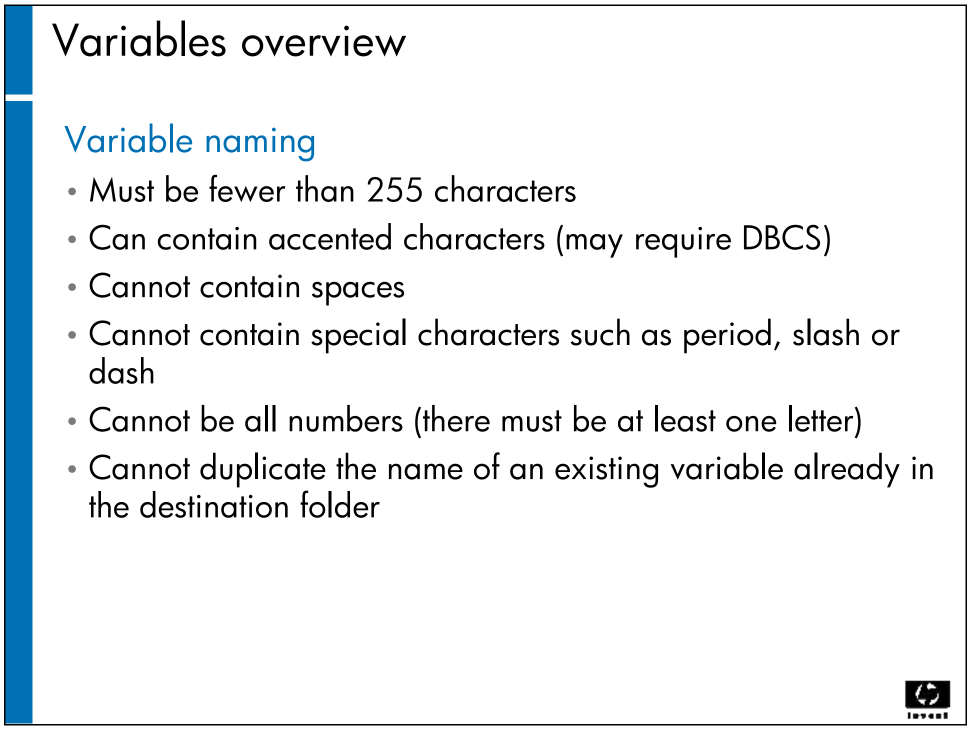
View Data Dictionary in Edit Panel (Portion)



Name	Description	Folder	Type	Source	Compute Time	Arr…	La…	Us…
SYS_BlankPageInBreak	The total number of bl…	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BlankPageInQueue	The total number of bl…	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BlankPageTotalI…	The total number of bl…	Root Folder	Integer	System	Page	No	No	Any
SYS_BreaksInQueue	The number of breaks…	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BundleCustomer1	The current customer …	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BundleCustomer2	The current customer …	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BundleCustomer3	The current customer …	Root Folder	Integer	System	As-needed	No	No	Any
SYS_BundleCustomer4	The current customer …	Root Folder	Integer	System	As-needed	No	No	Any

Latest

The list above shows an example of the *variables* and some of their *properties*. Notice the detail available in the list.



Variables overview

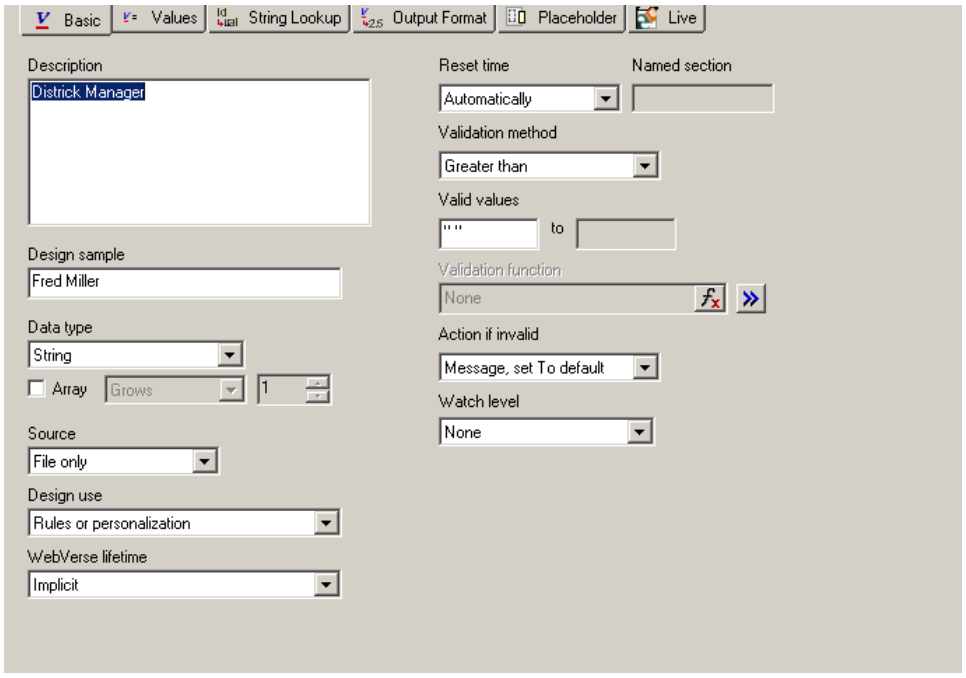
Variable naming

- Must be fewer than 255 characters
- Can contain accented characters (may require DBCS)
- Cannot contain spaces
- Cannot contain special characters such as period, slash or dash
- Cannot be all numbers (there must be at least one letter)
- Cannot duplicate the name of an existing variable already in the destination folder

As with other objects in the Library, both the **Description** and **Design Sample** properties are *optional*.

2.3 Variable Properties – Basic Tab

Variable Properties – Basic Tab



Description: Districk Manager

Design sample: Fred Miller

Data type: String

Array: Grows, 1

Source: File only

Design use: Rules or personalization

WebVerse lifetime: Implicit

Reset time: Automatically

Named section:

Validation method: Greater than

Valid values: " " to

Validation function: None

Action if invalid: Message, set To default

Watch level: None

Data type

You must specify the type of data this variable contains. Select one of the following from the **Type** drop-down list.

- **String –** the value of the variable contains text or a series of one or more characters. This can be a language-dependent setting.
  Limit: The size of string variables in Dialogue is 64K.
- **Integer –** value of the variable is a whole number.
  Limit: minus 2 billion to plus 2 billion.
- **Boolean –** value of the variable is a true or false value.
  Limit: dictated by available formats.
- **Floating –** value of the variable is a numeric value that includes decimals.
  Limit: Mantissa and exponent (mantissa is approximately 18 digits).
- **Date –** value of the variable is a date/time. Formatting may change automatically based on the customer's language.
  Limit: dictated by available formats.
- **Currency –** value of the variable is a monetary amount.
  Limits: -922337203685477.5808 to 922337203685477.5807. This is up to 14 places to the left and up to four to the right of the decimal place.

The following two options are available only with the **Dynamic Content Import Module** installed:

- **Placeholder –** the variable is replaced by a text or image file from an external directory during an Engine run.
  Limit: 255 characters.
- **Tagged Text –** the variable contains a string of text surrounded by formatting tags.
  Limit: String limits apply.

The default option in the **Type** drop-down list is **String**.

Array

With an array, a single variable represents multiple values. Arrays are often used for transaction data (such as all the monthly charges or credits for a credit card) or for constant values (such as the months in a year, quarter, or semester).

In a data file, the elements of an array might appear horizontally in a row of data, or vertically with each element in a separate row of data (record).

- If you select **Array**, specify in a drop-down list if the array **Grows** or is **Static**.
- The array type of **Grows** has no set limit to the number of elements in the variable.
- The array type of **Static** is a variable with a set number of elements. When you select this option, a box becomes active so you can enter the number of elements in the static array.
- A **Source** of **Counter** *cannot* be an array.

Source

In the **Source** property you direct Dialogue where the value(s) for the variable originates. Select one of the following from the drop-down list:

- **User value –** You are defining the values. Type a value in the **Initial value** text box on the **Values** tab.
- **Formula –** A formula computes the value. Dialogue calculates the value based on the formula you define in the **Formula** text box on the **Values** tab.
- **File only –** If the value comes from a data file, such as a Customer Driver File, Reference File, or Initialization File.
- **Crossref page number** – The value represents a cross-reference page number. Only used with the **Publication support** module.
- **Counter –** The value increments during an Engine run. For example, you may want to compute the total weight of all documents by summing the weight of each document and then place these variables on a summary page or custom report.

Design Use

In the **Design use** drop-down list, specify how you want to use the variable from the following options:

- **None** - If you do not want this variable available to users. It then becomes restricted and is not available to users for use in rules or design objects. It will not appear in variable lists available to users.
- **Can only be used to personalize -** To allow users to use this variable for personalizing their design objects, such as messages and pages.
- **Can only be used in rules -** To allow users to select this variable to create rules on design objects.
- **Rules or personalization -** To allow users to use this variable for any purpose.
- **Selector** – Can be used to create selection objects in the Live Editor.

Design Sample

Enter the text string that you want to display when this variable appears in text boxes, sections, and tables in Designer.

Reset time

Your choice in the **Reset time** drop-down list determines when Dialogue seeks information to use as values. This property is unavailable with the **Source** of **Counter**. The following options appear for all other **Sources**.

- **Automatically –** this option is the default selection. The Engine resets the variable before each customer and before it reads any sections (with section-based data).
- **Before each customer –** The Engine resets the variable before each customer is read, regardless of section-based data.
- **Never reset –** Dialogue carries forward the variable values and will not reset at the start of a new customer. Select this option for values that must remain the same for all customers.
- **Named section –** The Engine resets the variable before the *named section* in the data file. This enables you to create subtotals before a particular named section in the data file. If you select this option, you must type the section name with the exact same spelling and case you use throughout the application in the adjacent text box.
- **All sections –** Resets the variable before all sections, regardless of name.

Validation method

You can have Dialogue automatically validate that a proper value has been set for a variable. When you define a **Validation method**, you specify the type of validation you want Dialogue to perform, the validation range, and the actions to perform if the value is set to an invalid value.

From the drop-down list you choose one of the following:

- **None –** you do not want to validate the value of a variable when it changes.
- **Value range –** specify a range of valid values for the variable.
- **Length range –** specify a valid length range for the value.
- **Equal to –** specify a precise (valid) value for the variable.
- **Greater than –** specify a value the variable should be greater than.
- **Greater than or equal to –** specify a value the variable should be greater than or equal to.
- **Less than –** specify a value the variable should be less than.
- **Less than or equal to –** specify a value to which the variable should be less than or equal to.
- **Not equal to –** specify a precise value that is not valid for the variable.

- **Function** – A boolean function that returns true if validation succeeds. It refers to the data entered via **SYS_ValidationVariable**. If it sets the value of **SYS_ValidationVariable**, the variable being validated will be updated. If the function cannot be located or returns an error or a severe error, the validation fails.

Validation values

If you select anything other than **None** from the **Validation method** drop-down list, type the valid value or range of values in the **Valid values** fields or add validation function.

Action if invalid

Select one of the following from the **Action if invalid** drop-down list to specify the type of error processing to perform if the value is invalid:

- **Message, continue –** issue a message and continue processing.
- **Message, set To default –** issue a message, reset the variable to its default value, and continue processing.
- **Message, stop –** issue a message and stop processing.
- **Message, skip document –** skip the document if the variable is not valid or not found in the data.

When you define variables for validation, the **SYS_CustInvalidDataLevel** system variable sets based on the **Action if invalid** option chosen. The settings are:

0 – Variable is valid.

1 – Issues a message and sets the variable to default, and continues processing.

2 – Issues a message and halts processing.

10 – Issues a message and skips the customer.

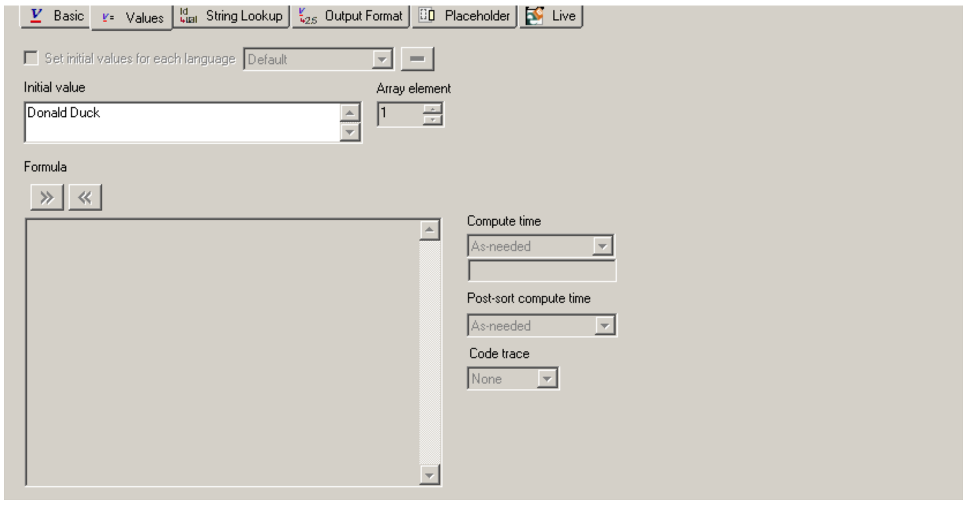
Watch level

You can set a **Watch level** on this variable to troubleshoot an Engine run. When the Engine runs, Dialogue writes information about the variable's processing in a debug file. Choose an option from the following:

- **None –** Avoid writing information about this variable.
- **Changed –** Write information each time the value changes. This choice gives you the most detail in the debug file.
- **Set –** Write information when the Engine sets the variable.

2.4 Variable Properties – Values Tab

Variable Properties – Values Tab



The options on the **Values** tab become active when you select **User value, Formula, File only**, or **Counter** from the **Source** drop-down list on the **Basic** tab. Options vary by **Source**.

Constant User Values

- In the **Initial value** text box on the Values tab, specify an initial variable value, up to 65,536K (32,766 for DBCS) characters. You can set a different value for each element in any array variable by choosing an **Array element** number before you type a value.
- If you use language layers with this variable, select the **Set initial values for each language** check box. Specify a separate **Initial value** for each language in the drop-down list.

File only

- When you select a **Validation method** on the **Basic** tab, the **Initial value** text box can be used to specify a default value for this variable.

Formula and Counters

The remaining properties on the **Values** tab become active when **Formula** or **Counter** is the variable source in the **Basic** tab.

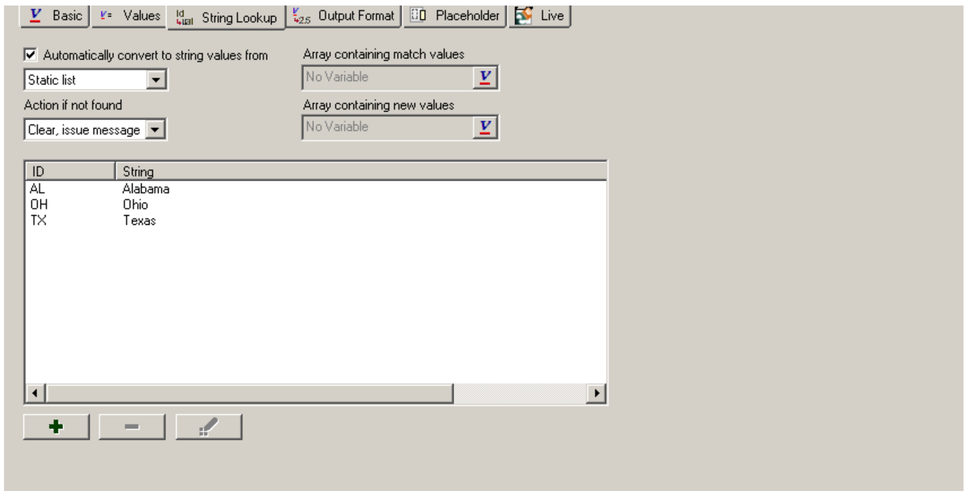
Formulas can be used to:

- Compute values based on data, other variables, or both
- Collect non-array information into arrays
- Perform subroutines
    - Conditional actions
    - Loops
    - Case logic
    - Functions

You will learn detailed information about formulas in a later chapter in this course.

2.5 Variable Properties – String Lookup Tab

Variable Properties – String Lookup Tab



☑ Automatically convert to string values from: Static list

Action if not found: Clear, issue message

Array containing match values: No Variable

Array containing new values: No Variable

ID	String
AL	Alabama
OH	Ohio
TX	Texas

String Lookup

With the **String lookup** option Dialogue replaces one string value with another value that you define. In the example above, if the value of a **CustomerState** variable is as "OH," Dialogue will substitute "Ohio" on the printed page.

String lookup is available if the **Data type** of a variable is **String** and the **Source** is **Formula, File only,** or **Counter**. To activate the String lookup option, select the **Automatically convert to string values from** check box. Select an option from the drop-down list to specify where the lookup values are located:

- **Static list** – The substitution list is stored locally.
- **Array variables** – The substitution values are stored in an array variable. There are two array variables: an array containing the match values and an array containing the replacement values.
- **Sorted array variables** – The string resides in an array variable. If the list is sorted, Dialogue will process the substitution more efficiently.

Action if not Found

You choose an option in the **Action if not found** drop-down list to specify what Dialogue does if it encounters an identifier that has not been defined.

- **Clear -** ignore the variable and continue processing.
- **Keep** - retain the value found in the data file and continue processing.
- **Clear, Issue message -** ignore the variable, issue a message, and continue processing.

- **Keep, Issue message** - retain the value found in the data file, issue a message, and continue processing.
- **Clear, generate error** - ignore the variable and stop processing.
- **Keep, generate error** - retain the value found in the data file and stop processing.

If you choose Static List

If you choose Static list from the drop-down list, the large text box and buttons become active.

- **Add a new automatic lookup value** – click the button. The **Lookup String** dialog box opens. On this dialog box, specify the **Identifier** Dialogue will encounter during processing, and the **Value** used to replace the **Identifier**.
- **Delete the selected lookup value** – highlight the string to delete and click the button. The lookup string is deleted and the list adjusts accordingly.
- **Edit the selected lookup string** – highlight the string to edit and click the button. The **Lookup String** dialog box opens. Make your changes and click **OK**.

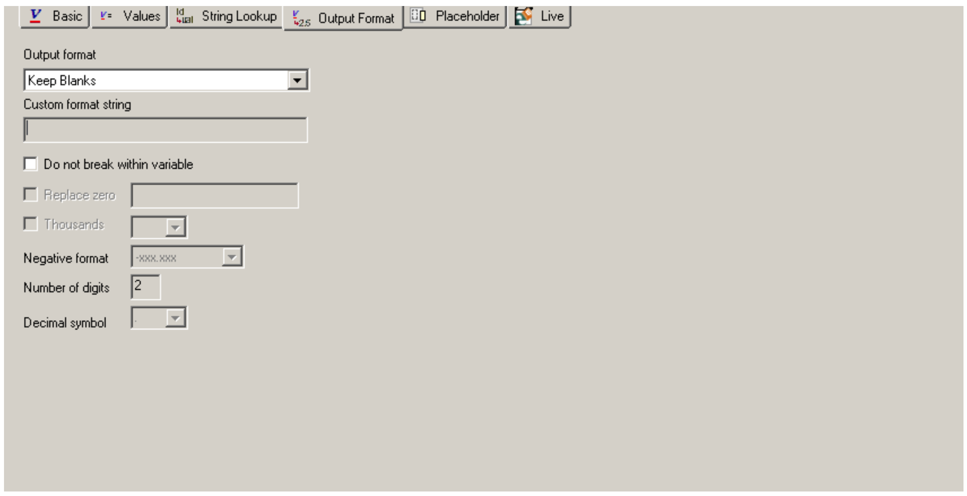
If you choose Array Variables

If you select **Array variables** or **Sorted array variables** from the drop-down list, two variable boxes become active.

- **Array containing match values** – search the specified array to match an identifier in a data file
- **Array containing new values** – substitute the new value at the same element number as the match value.

2.6 Variable Properties – Output Format Tab

Variable Properties – Output Format Tab

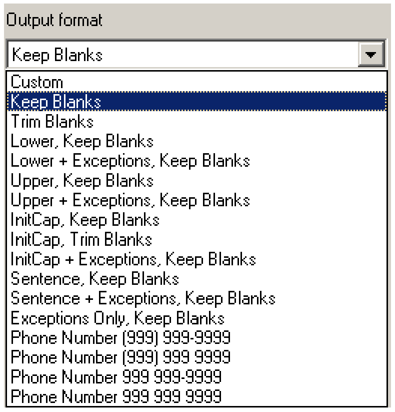


Output format: Keep Blanks
Custom format string:
☐ Do not break within variable
☐ Replace zero
☐ Thousands
Negative format: -xxx.xxx
Number of digits: 2
Decimal symbol: .

Output Format

The **Output format** drop-down list enables you to control how the data appears on the final composed page. The options available on the drop-down list vary depending on the **Data type** of the variable.

Output Format – Sample Date Formats



Output format: Keep Blanks

- Custom
- Keep Blanks
- Trim Blanks
- Lower, Keep Blanks
- Lower + Exceptions, Keep Blanks
- Upper, Keep Blanks
- Upper + Exceptions, Keep Blanks
- InitCap, Keep Blanks
- InitCap, Trim Blanks
- InitCap + Exceptions, Keep Blanks
- Sentence, Keep Blanks
- Sentence + Exceptions, Keep Blanks
- Exceptions Only, Keep Blanks
- Phone Number (999) 999-9999
- Phone Number (999) 999 9999
- Phone Number 999 999-9999
- Phone Number 999 999 9999

Custom Format String

- You can create a customized output format. When you choose **Custom** from the **Output format** drop-down list, the **Custom format string** text box becomes active. Use this text box to type a custom format.
- There are custom formats already available. To access the custom formats, right-click inside the **Custom format string** text box and choose **Format** from the shortcut menu.

Do Not Break Within Variable

Select the **Do not break within variable** check box to prevent the variable value from being split onto multiple lines when substituted into a paragraph. This check box is cleared by default.

Numeric variables that are formatted with spaces within them, such as for the thousands separator or tabs within the format, are automatically set to **Do not break within variable**.

When you use **Do not break within variable**, remember all paragraphs containing the variable must have room to include the full value on one line.

The **Do not break within variable setting** applies only to the variable itself. If you apply additional formatting to the variable, such as custom formatting to a chart legend or wrapping within a table cell, the custom formatting can cause the variable to break in the output.

Replace Zero

You can select to override the formatting of a variable if the value is zero. The override value can be any string. This can be set here in the variable properties or in the locale properties.

Thousands

For the Thousands separator, you can select a period, comma, or space. You can also enter a custom thousands separator in the box.

Negative Format

You can select how negative numbers are formatted. Options on the Negative format drop-down list are:

-xxx.xxx	(xxx.xxx)
- xxx.xxx	<xxx.xxx>
xxx.xxx-	< xxx.xxx >
xxx.xxx -	xxx,xxx CR
(xxx.xxx)	cr xxx,xxx

The display of currency symbols and decimal values are determined by the current locale and other output format settings described below.

Number of Digits

In the Number of Digits box, select a number of digits to display after the decimal point. For a Fixed Decimal format, it represents the number of digits that must display. For a Significant Decimal format, it represents the maximum number of decimal places to display. Two is the default number of digits.

Decimal Symbol

For the Decimal symbol, you can select either a period or comma; you can also enter a special character to act as the decimal symbol.

2.7 System Variables

The following pages summarize some key system variables.

Often Used

SYS_DateCurrent – Today's date: when the Engine runs to produce documents. Arguably this is the system variable used the most.

SYS_DateAsOf – The date a run completed. You can refer the Engine back to a run that completed at a specific time in the past.

SYS_DocPrintedValue – The document or chapter number that should be printed on the current page. This number increments with each document included for a given customer. This numbering restarts:

- At the beginning of the next customer, or
- When the Engine composes documents that specify a restart

This variable is often used with **SYS_PagePrintedValue**.

SYS_DocumentNames – An array of document names selected for this customer.

SYS_PageInDocument – A running page number for the current page. This variable does not reset at the start of a new document. If you use duplex pages and chose **Fronts and allowed backs** on the application properties, this variable counts both the front and back of a sheet. Also, banner pages are not counted with this variable.

SYS_PagePrintedValue – The page number that should be printed on the current page. This variable can reset at the start of a new document. Banner pages are not counted with this variable.

SYS_PageTotalInDocument – The total number of pages that were counted in the current document.

SYS_PageFlows – You set this system variable for each page to identify whether the page being processed has information that flows to another page or not. This variable is useful to determine when a message such as "continued…" should be placed on a page.

SYS_PagePhysicalInDocument – The current number of physical pages within the document being written. See **SYS_PageTotalInDocument** for the total counted pages in the document. The physical pages may differ from the counted pages based on page counting method set in the application.

SYS_PageTotalPhysicalInDocument – The total number of physical pages within the document being written. See **SYS_PageTotalInDocument** for the total counted pages in the document. The physical pages may differ from the counted pages based on page counting method set in the application.

SYS_LanguageCustomer – The current customer's preferred language. This is used to pick the correct language version of pages, messages, and text preferences, such as the names of days and months.

SYS_LocaleCustomer – The current customer's locale. This is used to determine the type of formatting for numbers, currency, and dates.

Customer Handling

SYS_CustomerInRun – The current customer number being processed.

SYS_CustInvalidDataLevel – The maximum invalid data level for the current customer, normally based on the action set in the variable's validation: 0 = no invalid data, 1 = continue (set default), 2 = error. If the value is greater than or equal to 10, the current customer is skipped, given that output to the customer has not already begun. The value of **SYS_CustInvalidDataLevel** can change when it is fired at different compute times.

SYS_CustomerChecksum – The checksum value of text and images in this document or customer.

SYS_CustomerEffectiveDate – The as-of date that documents should be created for this customer (used only for multi-version packages with Effectivity).

SYS_CustomerJurisdiction – The name of the jurisdiction associated with the current customer.

Campaigns

SYS_CampaignFirstPage – The printed page number on which this campaign begins. This can be used in campaign teasers to reference in a "See page number xx" tag line.

SYS_CampaignPriority – The relative priority for a campaign if set by a campaign priority property.

SYS_CampaignsQualified – The total number of campaigns that the current customer qualifies to receive.

SYS_CampaignsSent – The number of campaigns that were sent to the current customer.

SYS_MktgPagesAdded – The number of marketing pages included in a customer document by the Engine. These are pages created only because campaign messages caused generation of extra pages.

SYS_MktgPagesAllowed – The maximum number of marketing pages that can be added to the current customer. This is controlled in the application properties.

Messages

SYS_MsgContents – An array containing the actual contents of all currently sent messages.

SYS_MsgIdentifier – An array containing the message identifiers of all currently sent messages.

SYS_MsgName – An array containing the names of all currently sent messages.

Tables

SYS_TableRow – This variable outputs the current row count for an automated table row, including those that did not output.

SYS_TableRowOnPage – This variable increments with each composed row on the page that has the system variable. If the variable is not included in the header (for example), then the header is not counted. (If it included in the header, then the header *will* be counted.) For an automated row that is printed multiple times, the system variable increments each time the Engine composes the row. The counter resets at a page break.

SYS_TableRowPage – Unlike the previous variable that must be specifically placed on a row for a row to be counted, this variable outputs the current row for a table on the page with only the following (fixed) exceptions: headers, footers, or rows that are excluded due to rules. The counter resets at a page break.

SYS_TableRowTotal – This variable outputs the current row count for the table on all pages. The row count does not include headers, footers, or rows that were excluded because of rules.

Queues and Output Sorting

SYS_BreaksInQueue – The number of breaks (subsets) processed in the current output queue.

SYS_BundleCustomer*n* – The current customer number within a bundling tier (from 1 to 5). You can use this variable to control an inserter by household.

SYS_BundleCustomerTotal*n* – The total number of customers for a bundling tier (from 1 to 5). This variable can signal the end of a bundle.

SYS_BundleinBreak*n* – The bundle sequence number within the convenience break for a bundling tier (from 1 to 5).

SYS_BundleinQueue*n* – The bundle sequence number within the queue for a bundling tier (from 1 to 5).

SYS_BundlePage*n* – The current page number in a bundling tier (from 1 to 5).

SYS_BundlePageTotal*n* – The total page count in a bundling tier (from 1 to 5).

SYS_BundleSheet*n* – The current sheet count in a bundling tier (from 1 to 5).

SYS_BundleSheetTotal*n* – The total sheet count in a bundling tier (from 1 to 5).

SYS_ByteInBreak – The total number of bytes written to the current break.

SYS_ByteInQueue – The total number of bytes written to the current queue.

SYS_DocumentInQueue – The current document number being written into the current queue.

SYS_DocumentTotalInRun – The total number of customers that have been written to all output queues.

SYS_InserterInQueue – The user-specified name of an inserter associated with the current output queue.

SYS_NumBannersInQueue – The Number of banner pages that have been written to the queue so far.

SYS_PackageRequiredQueue – The required contents of the package file. This variable is used for reporting only.

SYS_PageInQueue – The total number of physical pages within the current queue. This variable will count banner pages.

SYS_QueueCurrent – The user-specified name of the queue for the current document.

SYS_QueueFileName – The name of the current output file for the current output queue.

SYS_SheetInQueue – The current sheet number in the current output queue.

SYS_SortIndex – Must be included in a Sort Index file so the Engine can find corresponding data in the Sort Data File (20 bytes long).

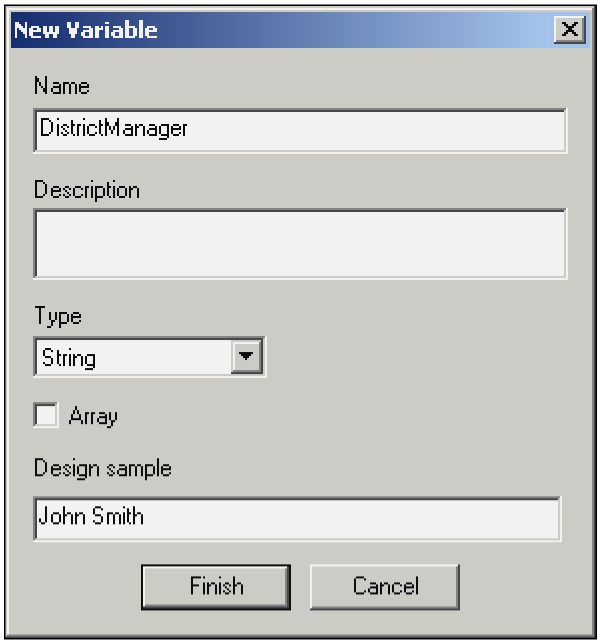
2.8 Exercise: Create Variables

Create a Variable with a default value

You will use the following variable later when you complete an exercise for the "Mapping Data Files" chapter.

1. In Design Manager, navigate to:
   Exstream > Exercise 02-05 Data Dictionary.
2. In the Exercise 02-05 folder, right-click the Data Dictionary heading.
3. Choose New Variable from the shortcut menu.
   The New Variable dialog box appears.
4. In the Name box, enter `DistrictManager`.
5. In the Type drop-down list choose String.
6. In the Design sample text box type `John Smith`.

Create a New Variable



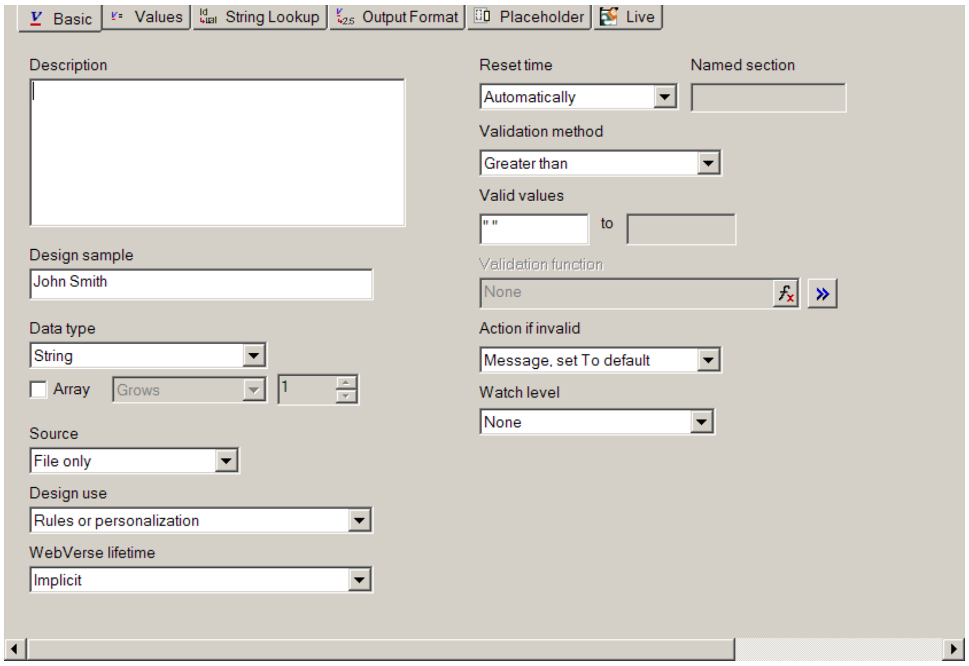
Name: DistrictManager
Description:
Type: String
Array: ☐
Design sample: John Smith

7. Click **Finish**.

The variable appears in the Property Panel for you to define.

8. On the **Basic** tab, in the **Source** drop-down list choose **File only**.
9. From the **Validation method** list box, select **Greater than**.
10. In the **Valid values** text box, type "".
11. In the **Action if invalid** list box, select **Message, set to default**.

Variable Properties - Basic Tab



Description

Design sample: John Smith

Data type: String

Array: Grows, 1

Source: File only

Design use: Rules or personalization

WebVerse lifetime: Implicit

Reset time: Automatically

Named section

Validation method: Greater than

Valid values: " " to

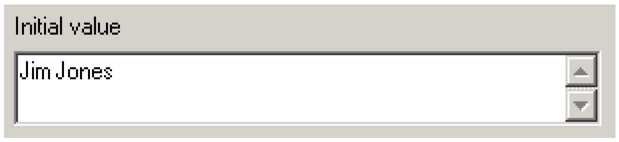
Validation function: None

Action if invalid: Message, set To default

Watch level: None

12. Click the **Values** tab.
13. In the **Initial value** text box, enter `Jim Jones`.

Variable Properties - Values Tab



Initial value: Jim Jones

14. Save your work and exit the Property Panel.

Create a Variable with a string lookup table

You will use this variable later when you complete an exercise in the "Mapping Data Files" chapter.

1. In Design Manager, navigate to:
   Exstream > **Exercise 02-05**.
2. In the **Exercise 02-05** folder, right-click the **Data Dictionary** heading.
3. Choose **New Variable** from the shortcut menu.
   The **New Variable** dialog box appears.
4. In the **Name** box, enter `StateName`.
5. In the **Type** drop-down list choose **String**
6. In the **Design sample** text box, type `Kentucky`.
7. Click **Finish**.
   The variable appears in the Property Panel for you to define.
8. On the **Basic** tab, in the **Source** drop-down list, choose **File only**.
9. Click the **String Lookup** tab.
10. Select the **Automatically convert to string values from** check box.
11. Click the + button
    The **Lookup String** dialog box appears.
12. In the **Identifier** Text box, type `AL`.
13. In the **Value** text box, type `Alabama`.
14. Click **OK**.
15. Repeat steps 11-14, adding the **Identifier/Value** pairs from the table below.

Identifier	Value
CO	**Colorado**
FL	**Florida**
IL	**Illinois**
KY	**Kentucky**
NC	**North Carolina**
OH	**Ohio**
OR	**Oregon**
RI	**Rhode Island**
TX	**Texas**

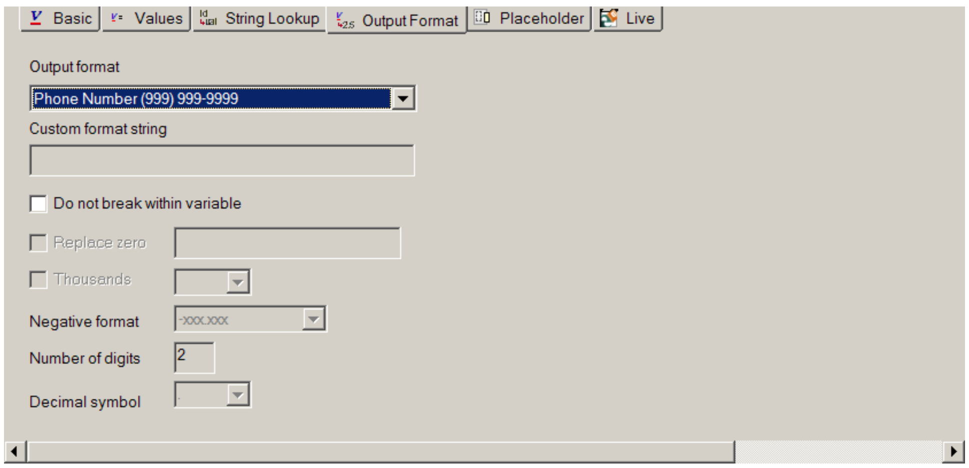
16. Save your work and exit the Property Panel.

Create a Variable with a default output format

You will use the following variable later when you complete an exercise for the "Mapping Data Files" chapter.

1. In Design Manager, navigate to:
   Exstream > **Exercise 02-05**.
2. In the **Exercise 02-05**, right-click the **Data Dictionary** heading.
3. Choose **New Variable** from the shortcut menu.
   The **New Variable** dialog box appears.
4. In the **Name** text box enter `CustomerCareNum`.
5. In the **Type** drop-down list choose **String**.
6. In the **Design sample** text box enter `(888) 888-8888`.
7. Click **Finish**.
   The variable appears in the Property Panel for you to define.
8. On the **Basic** tab, in the **Source** drop-down list, choose **File only**.
9. Click the **Output Format** tab.
10. From the **Output format** list box, select **Phone Number (999) 999-9999**.

Variable Properties – Output Format Tab



Output format: Phone Number (999) 999-9999
Custom format string:
Do not break within variable
Replace zero
Thousands
Negative format: -xxx.xxx
Number of digits: 2
Decimal symbol

11. Save your work and exit the property panel.

Chapter 3: Data Files

In this Chapter

This chapter discusses Dialogue Data Files. You will learn to create and define data files by type and format. You will also map variables to the data files.

Objectives

By the end of this chapter you will:

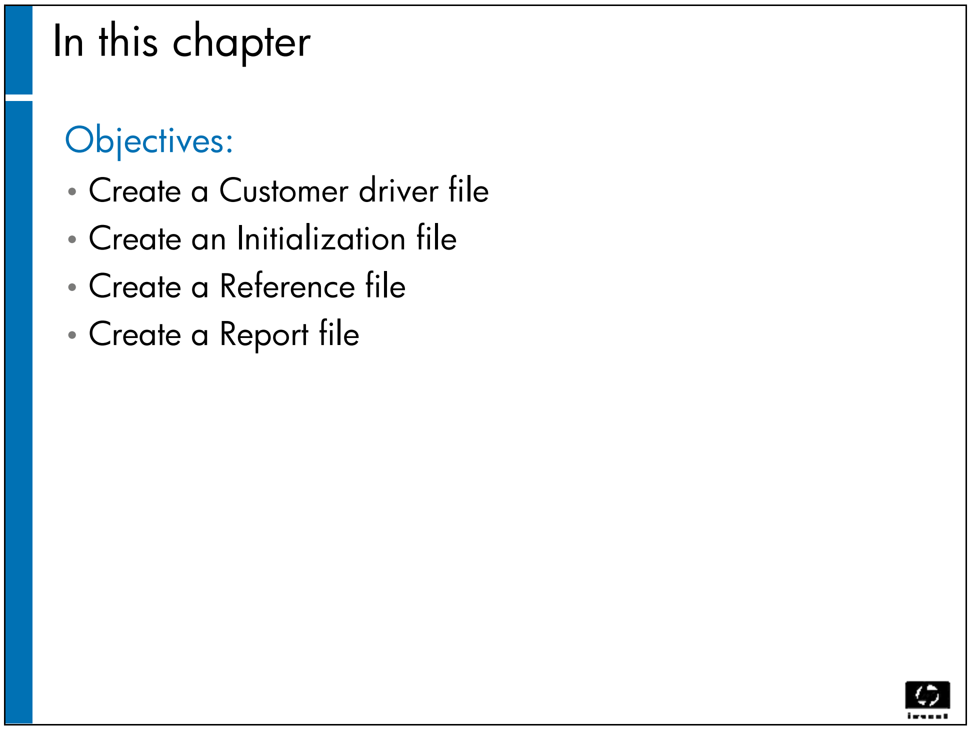
- Create a Customer driver file.
- Create an Initialization file.
- Create a Reference file.
- Create a Report file.

For more information

For more information, refer to the following Dialogue documentation.

Data Files guide	DataFile.pdf
System Administration guide	System.pdf

In this Chapter

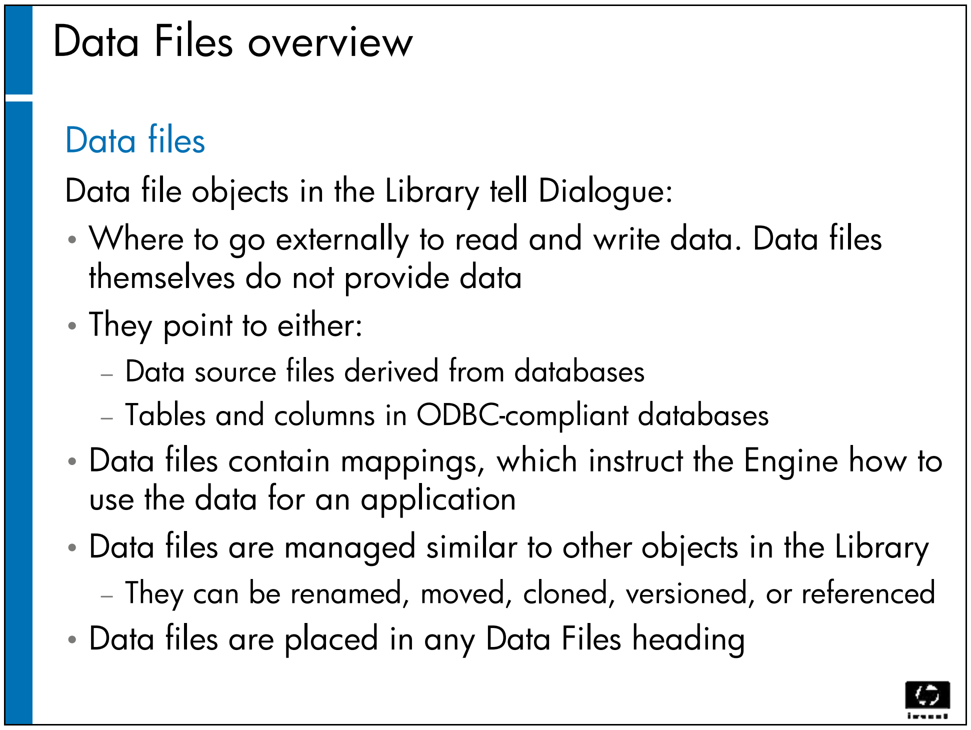


In this chapter

Objectives:

- Create a Customer driver file
- Create an Initialization file
- Create a Reference file
- Create a Report file

3.2 Data Files Overview



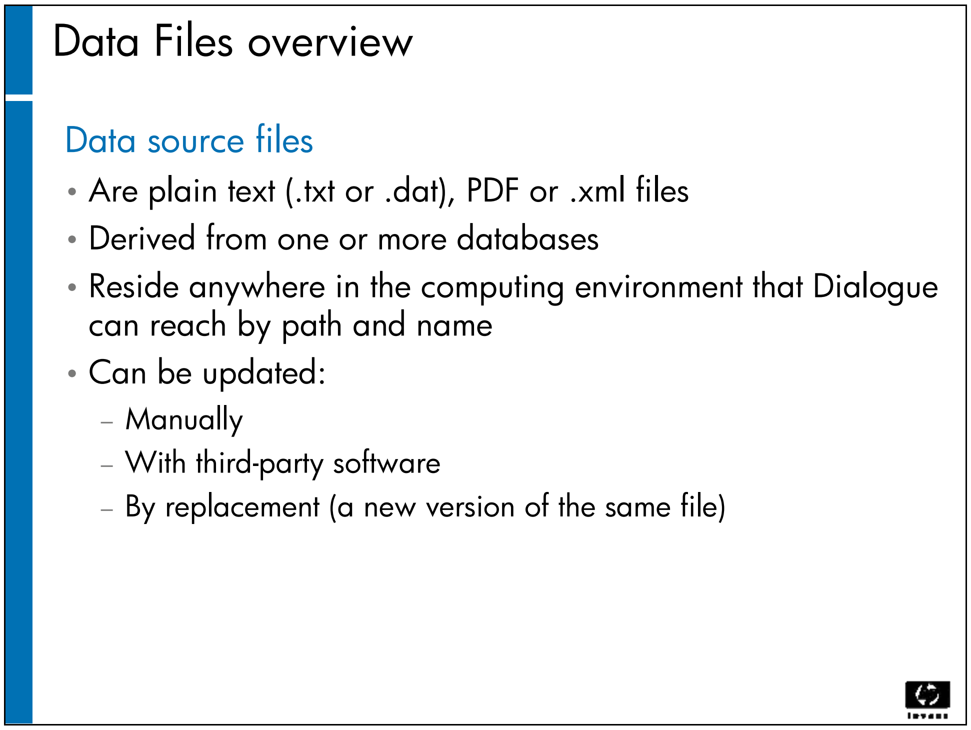
Data Files overview

Data files

Data file objects in the Library tell Dialogue:

- Where to go externally to read and write data. Data files themselves do not provide data
- They point to either:
  - Data source files derived from databases
  - Tables and columns in ODBC-compliant databases
- Data files contain mappings, which instruct the Engine how to use the data for an application
- Data files are managed similar to other objects in the Library
  - They can be renamed, moved, cloned, versioned, or referenced
- Data files are placed in any Data Files heading

Every application requires *data* to complete an Engine run. This section discusses the Library objects the Engine uses to find, read, and write data.

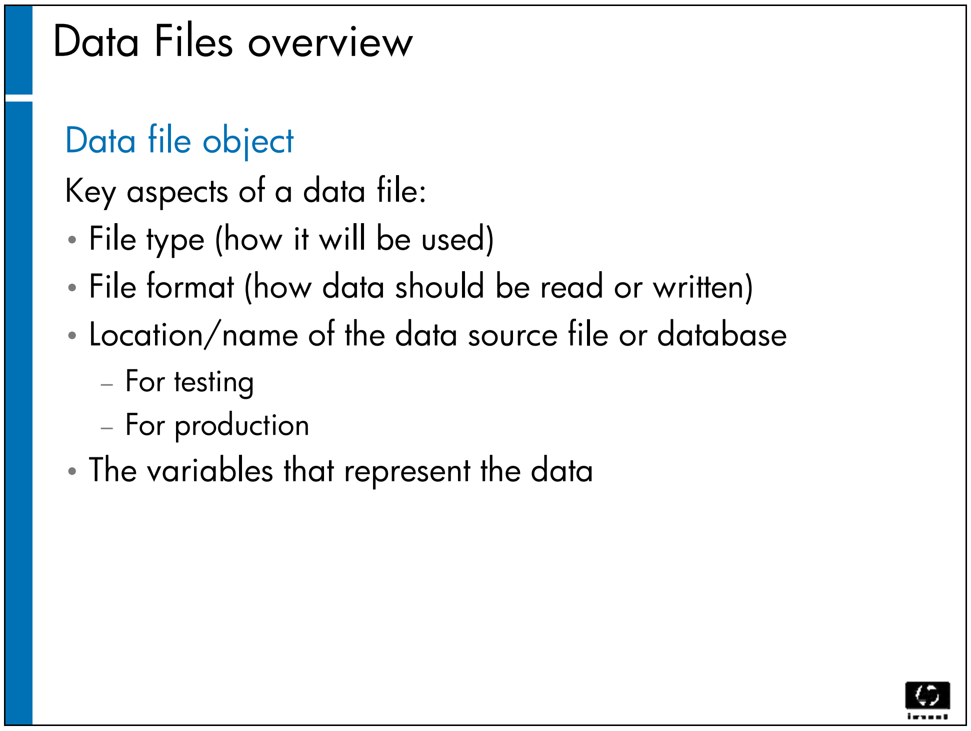


Data Files overview

Data source files

- Are plain text (.txt or .dat), PDF or .xml files
- Derived from one or more databases
- Reside anywhere in the computing environment that Dialogue can reach by path and name
- Can be updated:
  - Manually
  - With third-party software
  - By replacement (a new version of the same file)

Data Source Files are always external to Dialogue. Since they do *not* appear in the Library, you need to develop methods to keep track of these files: where they are in your computing systems, what data they contain, when their data needs to be updated, and so forth.



Data Files overview

Data file object

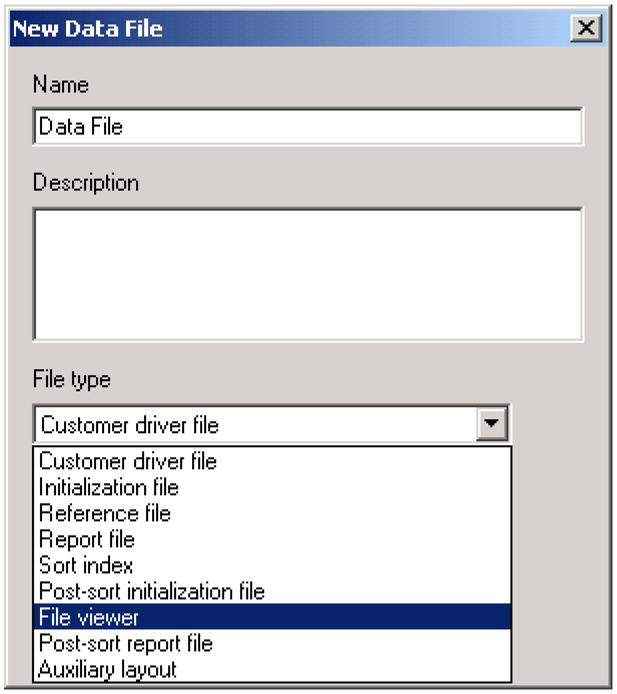
Key aspects of a data file:

- File type (how it will be used)
- File format (how data should be read or written)
- Location/name of the data source file or database
  - For testing
  - For production
- The variables that represent the data

3.3 File Type

When you create a Dialogue data file, your choice of **File Type** determines how the Engine will use the data file and the specific properties you see and access. Some options are tied to specific software modules you must have installed on your system.

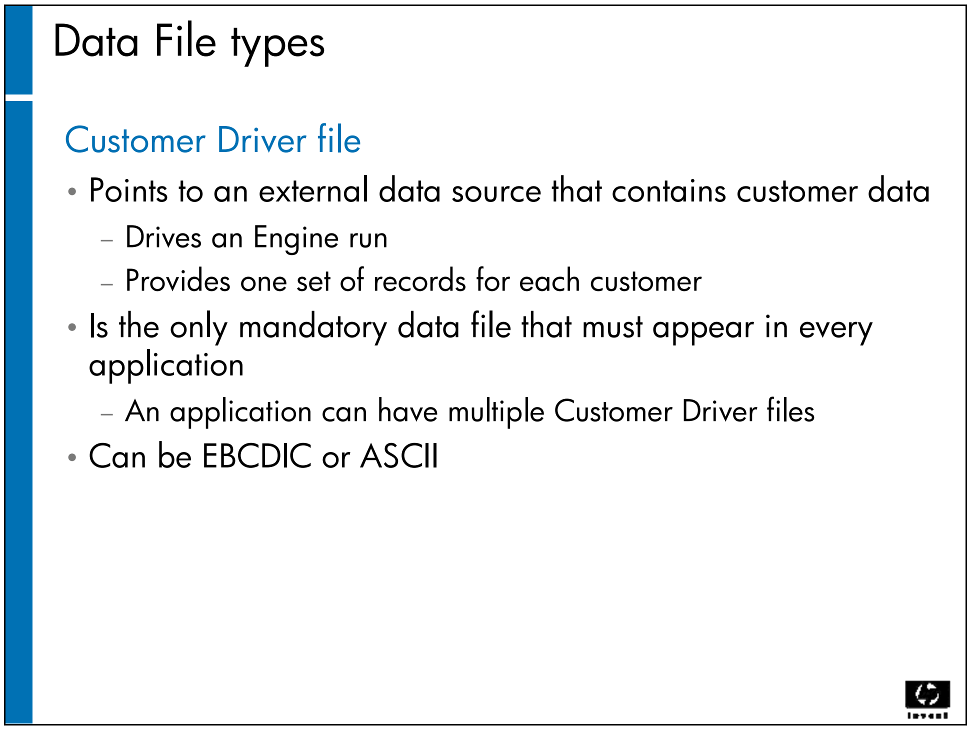
Data File Type Options in New Data File Dialog Box



Data File

Customer driver file

Customer driver file
Initialization file
Reference file
Report file
Sort index
Post-sort initialization file
File viewer
Post-sort report file
Auxiliary layout



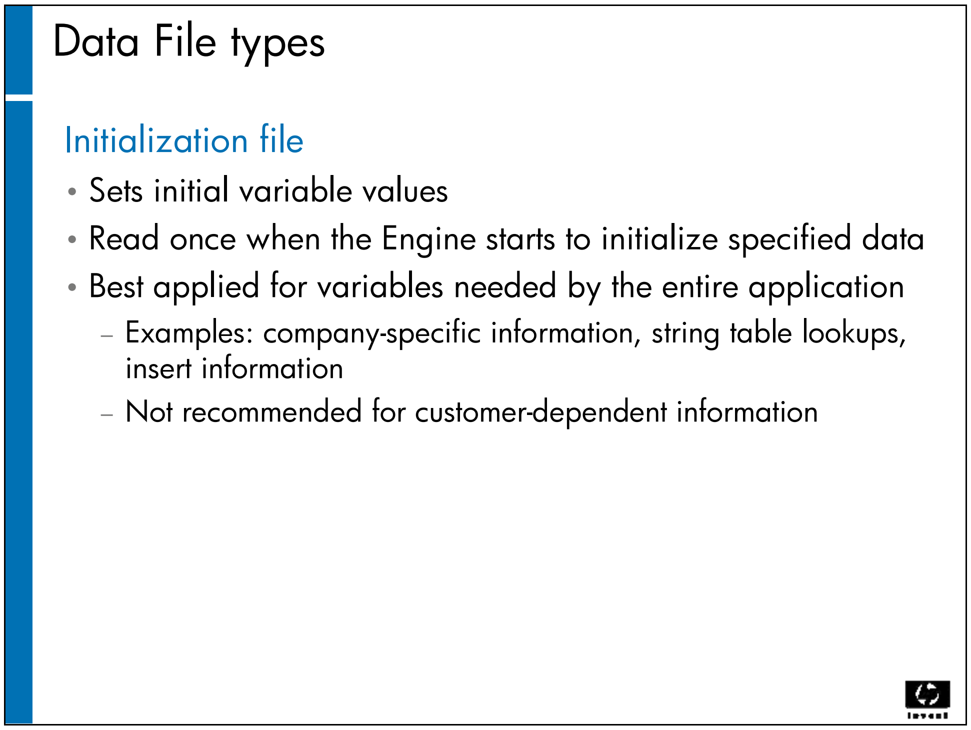
Data File types

Customer Driver file

- Points to an external data source that contains customer data
  - Drives an Engine run
  - Provides one set of records for each customer
- Is the only mandatory data file that must appear in every application
  - An application can have multiple Customer Driver files
- Can be EBCDIC or ASCII

Customer Driver Files can be:

- **Simple** – the data source file contains one set of data for each customer.
- **Section-based** – the data source contains multiple *sections* of data for each customer.

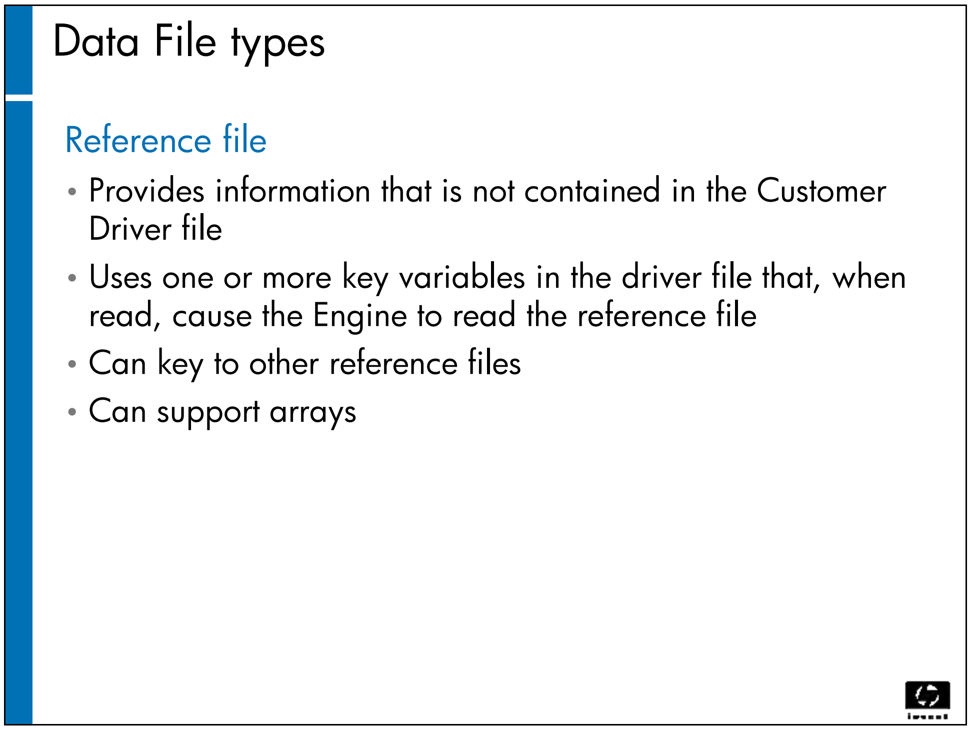


Data File types

Initialization file

- Sets initial variable values
- Read once when the Engine starts to initialize specified data
- Best applied for variables needed by the entire application
  - Examples: company-specific information, string table lookups, insert information
  - Not recommended for customer-dependent information

In the Library, Initialization Files must appear at the top of the list of **Data Files** under an application.



Data File types

Reference file

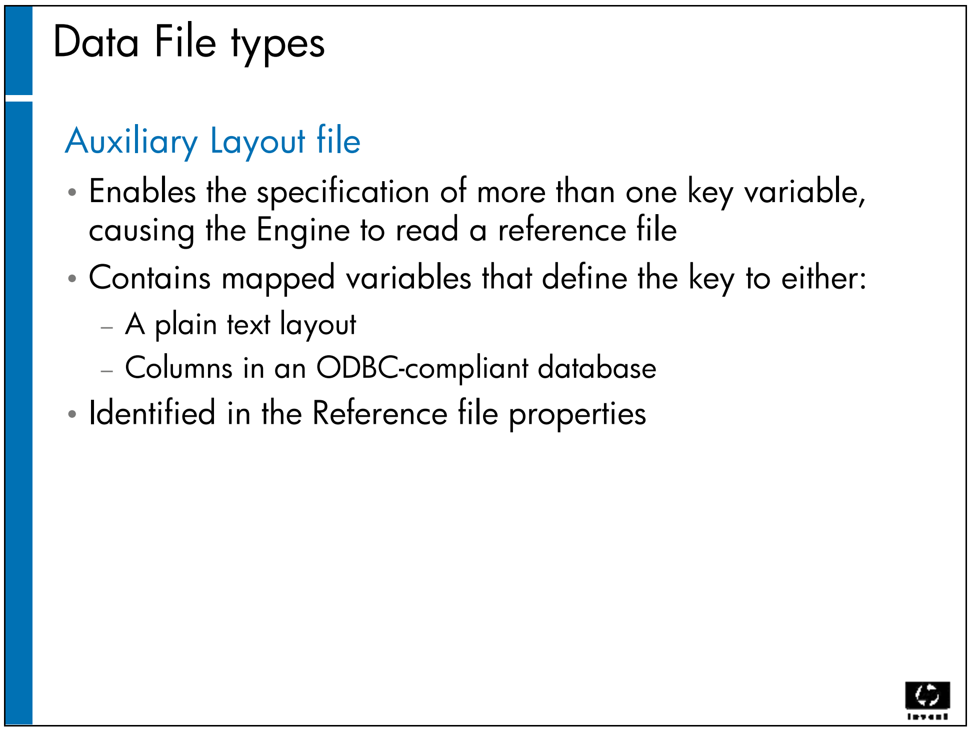
- Provides information that is not contained in the Customer Driver file
- Uses one or more key variables in the driver file that, when read, cause the Engine to read the reference file
- Can key to other reference files
- Can support arrays

A Reference file contains a key so that you can retrieve a set of data based on the

Engine reading one or more variables in the Customer Driver File. For example, a Reference File may contain the name and address of the store nearest to a customer, based on the customer's zip code.

An application can contain more than one **Reference** File. The Engine processes Reference files in the order they appear under the application in the Library.

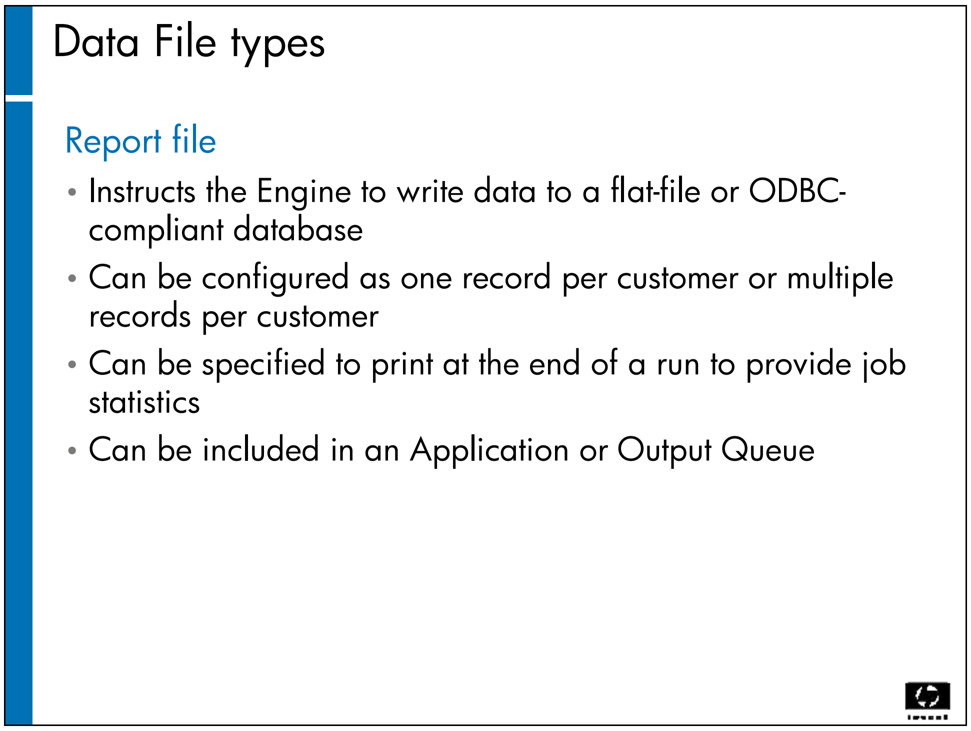
The properties of a Reference file limit you to one key variable. However, you can use multiple key variables by creating an **Auxiliary Layout** file.



Data File types

Auxiliary Layout file

- Enables the specification of more than one key variable, causing the Engine to read a reference file
- Contains mapped variables that define the key to either:
  - A plain text layout
  - Columns in an ODBC-compliant database
- Identified in the Reference file properties



Data File types

Report file

- Instructs the Engine to write data to a flat-file or ODBC-compliant database
- Can be configured as one record per customer or multiple records per customer
- Can be specified to print at the end of a run to provide job statistics
- Can be included in an Application or Output Queue

A Report file causes the Engine to *write* data. The other types of data files instruct the Engine to *read* data. You can use a Report file to produce an audit file, system control file, or other data reporting file.

Data Files with High-Volume Production

The Sort Index file, Post-Sort Report file, and Post-Sort Initialization file are only used with the Output Sorting and Bundling module of the High-Volume Production suite. These modules support two runs of the Engine for an application:

- With a Sort index file you can change the order, number, and other details about customers before you actually compose output for an application.
- These data file types are very similar to the standard Initialization files and Report Files. The principal difference is *timing*.

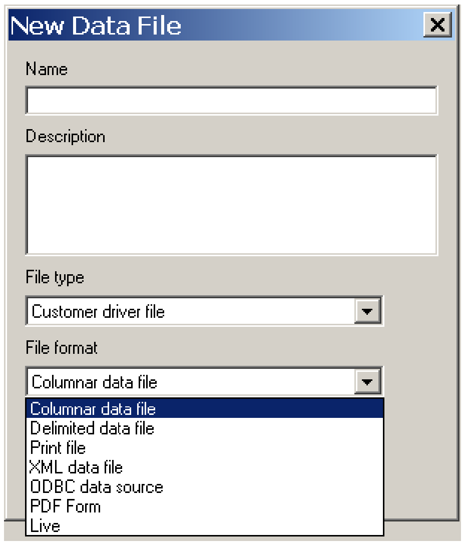
Viewer File

- You can view AFP or Metacode output files by defining *Viewer* data files in the Library.
- After an Engine run, you drag a Viewer file to the Edit Panel to view the composed print stream code.

3.4 File Format

When you create or update a Dialogue data file, your choice of **File Format** tells the Engine how to read/process data.

File Format Options in New Data File Dialog Box

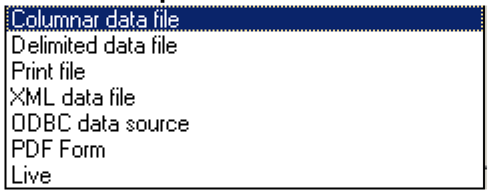
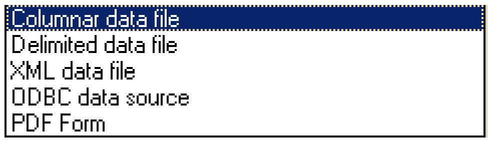


Note that the last five choices are all dependent upon specific software modules installed on your system.

Options

The options in the **File Format** drop-down list change according to the **File Type** you select.

File Format Options for Specific File Types

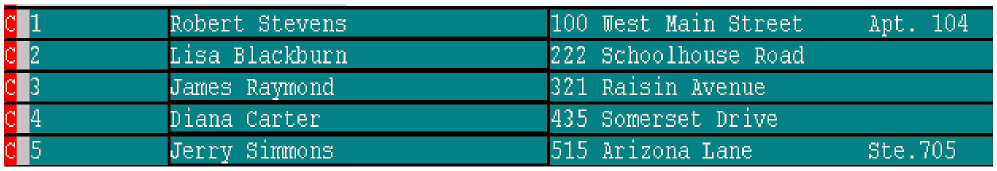
Data File Type	**File Format Options**
Customer Driver Reference	 Columnar data file Delimited data file Print file XML data file ODBC data source PDF Form Live
Initialization Report Post-Sort Initialization Post-Sort Report Auxiliary Layout	 Columnar data file Delimited data file XML data file ODBC data source PDF Form
Sort Index	Columnar data file
File Viewer	Metacode file AFP file

NOTE: If you change the **File Format** after a data file has been mapped, you will lose the entire data map. This will also occur if you change the record indicator type or length. Try to make these decisions before you begin to map the file.

Columnar Data File

A *columnar data file* points to a source file that has columns of data from the top to the bottom of the file. You can look straight down the column and easily see where a data area field begins and ends.

Columnar Data File in Edit Panel

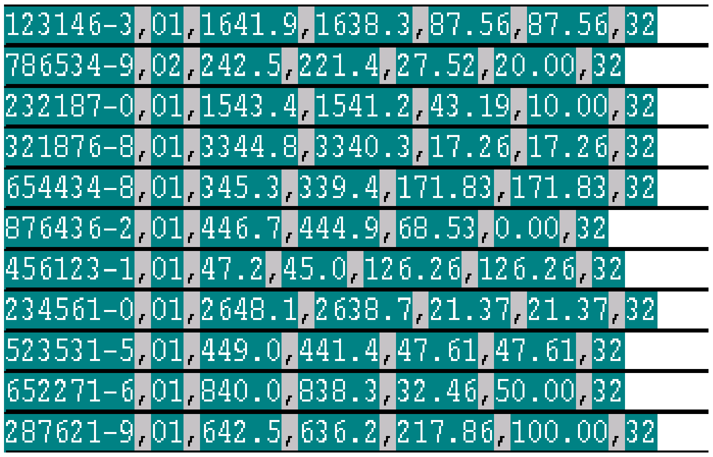


```
C 1           Robert Stevens          100 West Main Street     Apt. 104
C 2           Lisa Blackburn          222 Schoolhouse Road
C 3           James Raymond           321 Raisin Avenue
C 4           Diana Carter            435 Somerset Drive
C 5           Jerry Simmons           515 Arizona Lane         Ste.705
```

Delimited Data File

A *delimited data file* points to a source file that has fields separated by the same character. These characters mark the beginning and end of a data field. Often, these characters are inserted by your database software. A character may be ASCII- or hex-based.

Comma-Delimited Data File in Edit Panel



```
123146-3,01,1641.9,1638.3,87.56,87.56,32
786534-9,02,242.5,221.4,27.52,20.00,32
232187-0,01,1543.4,1541.2,43.19,10.00,32
321876-8,01,3344.8,3340.3,17.26,17.26,32
654434-8,01,345.3,339.4,171.83,171.83,32
876436-2,01,446.7,444.9,68.53,0.00,32
456123-1,01,47.2,45.0,126.26,126.26,32
234561-0,01,2648.1,2638.7,21.37,21.37,32
523531-5,01,449.0,441.4,47.61,47.61,32
652271-6,01,840.0,838.3,32.46,50.00,32
287621-9,01,642.5,636.2,217.86,100.00,32
```

Print File

With the **Print Miner** module, this data file points to a file created for a line printer as the data source.

XML Data File

Available with the **XML Input** module, this data file points to an external .xml file.

ODBC Data Source

Available with the **ODBC Access** module, this data file points to an ODBC-compliant database.

3.5 Exercise: Create Data Files

In this exercise, you will create data files. These data files will be defined and used in the next few exercises.

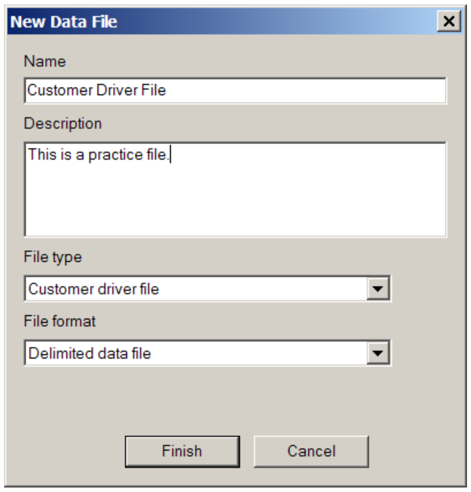
A. Create an Initialization File

1. Right-click the **Data Files** heading under the **Exercise 02-05 Data Files** folder.
2. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
3. Enter `Initialization File` in the **Name** text box.
4. Choose **Initialization file** from the **File type** drop-down list.
5. Choose **Columnar data file** from the **File format** drop-down list.
6. Click **Finish**.
   Dialogue adds your data file to the Library and displays the file in the Property Panel. You will define the properties for this data file in the next exercise.

B. Create a Customer Driver File

1. Right-click the **Data Files** heading under the **Exercise 02-05 Data Files** folder in the Library.
2. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
3. Enter `Customer Driver File` in the **Name** text box.
4. Enter `This is a practice file` in the **Description** text box.
5. Ensure **Customer driver file** appears in the **File Type** drop-down list.
6. Choose **Delimited data file** from the **File format** drop-down list.

New Data File Dialog Box



Name: Customer Driver File

Description: This is a practice file.

File type: Customer driver file

File format: Delimited data file

7. Click **Finish**.
   Dialogue adds your data file to the Library and displays the file in the Property Panel. You will define the properties for this data file in the next exercise.

C. Create a Reference File

1. Right-click the **Data Files** heading under the **Exercise 02-05 Data Files** folder.
2. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
3. Enter `Reference File` in the **Name** text box.
4. Choose **Reference file** from the **File type** drop-down list.
5. Choose **Delimited data file** from the **File format** drop-down list.
6. Click **Finish**.
   Dialogue adds your data file to the Library and displays the file in the Property Panel. You will define the properties for this data file in the next exercise.

D. Create a Report File

1. Right-click the **Data Files** heading under the **Exercise 02-05 Data Files** folder.
2. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
3. Enter `Report File` in the **Name** text box.
4. Choose **Report file** from the **File type** drop-down list.
5. Choose **Columnar data file** from the **File format** drop-down list.

6. Click **Finish**.
   Dialogue adds your data file to the Library and displays the file in the Property Panel. You will define the properties for this data file in the next exercise.

Chapter 4: Data File Properties

4.1 In this Chapter

This chapter discusses the various properties you can define for data files in the Property Panel.

Objectives

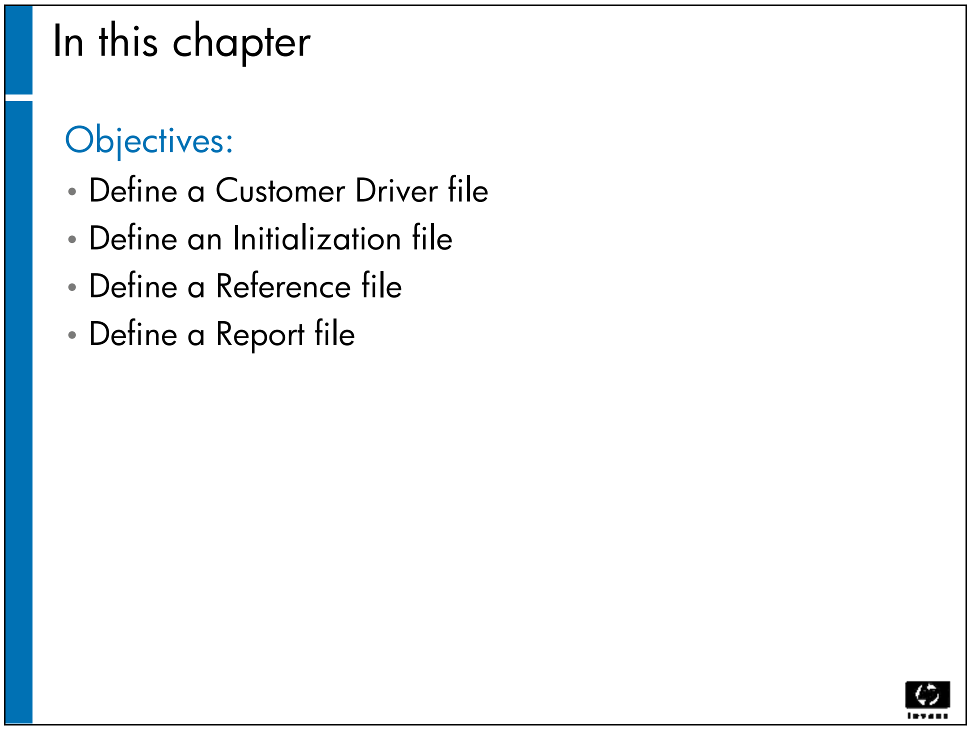
By the end of this chapter you will define the properties of the data files you created in the last exercise:

- Customer Driver File.
- Initialization File.
- Reference File.
- Report File.

For more information

For more information, refer to the following Dialogue documentation.

▪ *Data Files guide*	DataFile.pdf
▪ *System Administration*	System.pdf



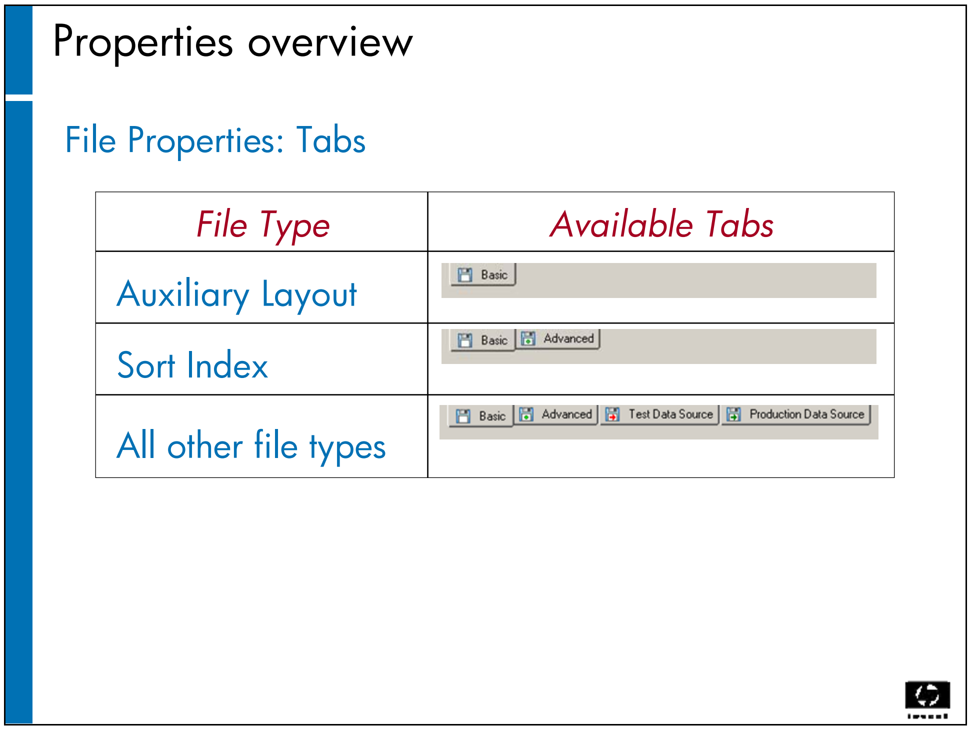
In this chapter

Objectives:

- Define a Customer Driver file
- Define an Initialization file
- Define a Reference file
- Define a Report file

4.2 Data File Properties Overview

You define most data file properties in the Property Panel. These properties differ, depending on your choice of **File Type** – including the availability of tabs, as shown in the next slide.



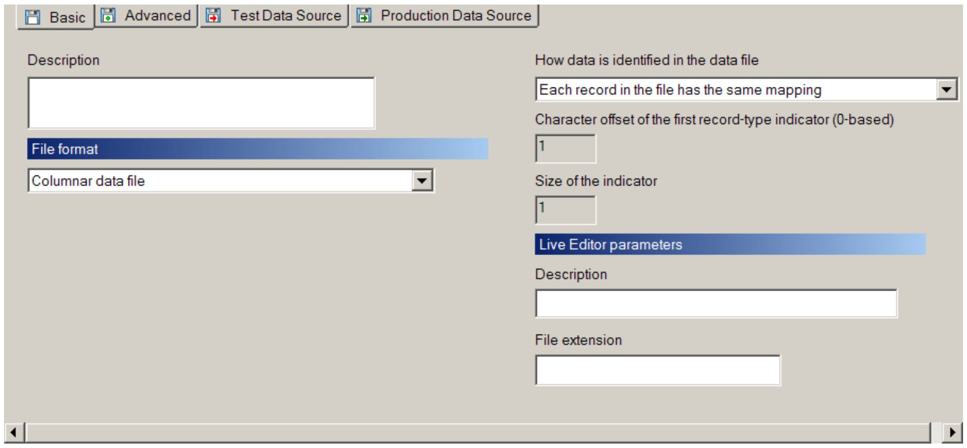
Properties overview

File Properties: Tabs

File Type	*Available Tabs*
Auxiliary Layout	Basic
Sort Index	Basic | Advanced
All other file types	Basic | Advanced | Test Data Source | Production Data Source

4.3 Basic Tab

Data File Properties – Basic Tab for Initialization File



Basic | Advanced | Test Data Source | Production Data Source

Description

File format

Columnar data file

How data is identified in the data file

Each record in the file has the same mapping

Character offset of the first record-type indicator (0-based)

1

Size of the indicator

1

Live Editor parameters

Description

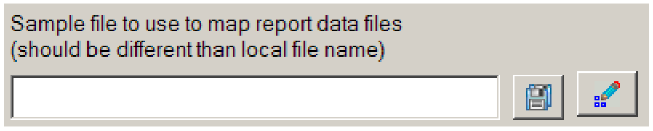
File extension

File Type Area: Identify Sample Layout File

Four of the **File Types** have you identify a *layout file* in the **File Type** area. As examples:

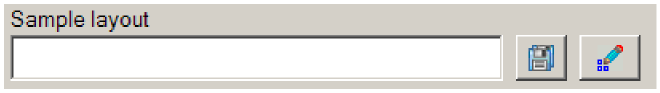
- A sample layout file for an Auxiliary Layout file maps all the variables that cause the Engine to read a Reference File.
- A sample layout file for the two types of Report Files tells the Engine the layout, variables, and static text to use to create a report.

Report File and Post-Sort Report



Sample file to use to map report data files
(should be different than local file name)

Auxiliary Layout and Sort Index



Sample layout

You click the 🔍 button to browse and select the file in the **Open** dialog box. The file's path and name appears in the text box.

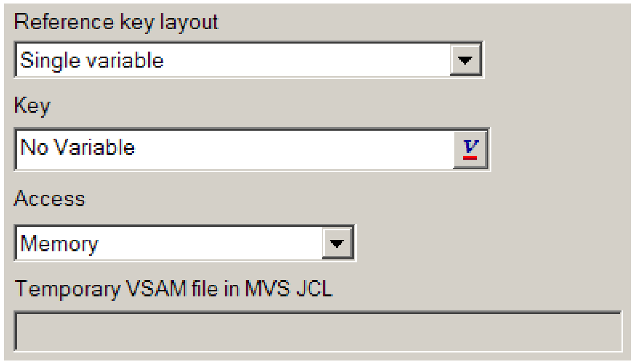
You can click the ✎ button to:

- Open the external file in your default editor to *update* its data.
- Open your default editor to *create* the file.

File Type Area: Reference File Options

When you choose **Reference File** as the file type, three **additional** options become available in the **File Type** area.

Reference File – File Type Area



Reference key layout
Single variable
Key
No Variable
Access
Memory
Temporary VSAM file in MVS JCL

Key: Single Variable

If a single variable has the Engine read the data mapped to this Reference file, you identify the variable that will be a unique **Key** for establishing a relationship between another file and this file.

For example, if you have customer data in a Customer driver file and transaction data in a Reference file. A variable in the Customer driver File you select as the **key**, such as **CustomerNumber** or **CustomerName**, causes the Engine to read corresponding information in the Reference file.

The key variable must have a unique value for each set of records in the Reference file and be available from, or computed from, data in the Customer driver file.

You can set this property to have a Reference file direct the Engine to another Reference file.

- Click the V button to select the variable you want to use as the **Key** to this file from the **Select Variable** dialog box.
- Leave the default of **Single Variable** in the **Reference Key Layout** drop-down list.

Key: Multiple Variables

If more than one variable has the Engine read the data mapped to this Reference file, you use the **Reference Key Layout** drop-down list. The variables that comprise the key must be contained in an existing Auxiliary Layout data file in the Library.

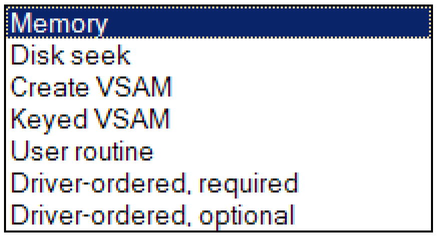
As with the Single Variable option, you can set this property to have a Reference file direct the Engine to another Reference file.

- Choose the Reference Key file name from the **Reference Key Layout** drop-down list. The other options in the list are all the data files in the Library that have the **File Type** of Auxiliary Layout.

The default is **Single Variable**, which permits you to use the **Key** text box. Once you select Auxiliary Layout, the **Key** text box becomes unavailable. You cannot use both properties at the same time.

Access

Access Options



The **Access** drop-down list enables you to choose a method for storing and re-accessing information from a Reference File.

- **Memory** – reads the entire Reference File and stores all the data in memory. This offers the fastest processing, but uses the most memory.
- **Disk Seek (Windows and Unix Only)** – reads the entire Reference File, keeps the reference keys in memory, and seeks the information on the disk as it is needed from the proper file position.
- **Create VSAM** – creates a temporary ESDS VSAM file. Used only for ZOS, MVS, and OS/390. Stores the keys in memory and seeks and reads the file for each reference, as in the **Disk Seek** option. You must enter the name of the temporary file in the **Temporary VSAM File in MVS JCL** text box. For more information, see the *More about Create VSAM* section.
- **Keyed VSAM** – uses an already existing KSDS VSAM file. Used only for ZOS, MVS, and OS/390. The key can be either numeric or string. The information is accessed as it is needed. No keys are stored in memory. For more information, see the *More about Keyed VSAM* section.
- **User Routine** – uses a routine created by a programmer at your site. This option is available with the **Dynamic Data Access** Module.
- **Driver-ordered, required** - The Engine reads customer data in the Customer driver file, looks for corresponding key variable values in the Reference file, and returns to the Customer driver file for the next customer. The Customer driver file and Reference file values must be in the same order. No keys are stored in

memory. If the Reference file does not have a match at the same customer position, then the Engine reads the remainder of the Reference file until it finds a match. If the Engine reaches the end of the Reference file without a match, the run stops and the Engine issues a severe message. This feature is supported only with columnar, delimited, and XML files.

- **Driver-ordered, optional** - The Engine reads customer data in the Customer driver file, looks for corresponding key variable values in the Reference file, and returns to the Customer driver file for the next customer. The Customer driver file and Reference file values must be in the same order. If the Reference file does not have match at the same customer position, the Engine immediately stops reading and resets the Reference file position to test for a match for the next customer. This feature is supported only with XML and columnar files.

More about Create VSAM

The **Create VSAM** option enables the data file to be translated into a VSAM file on MVS mainframe production systems. VSAM offers faster retrieval of data during lookup operations when you run the Dialogue Engine on MVS.

Additionally, a value is required to be entered into **Temporary VSAM file in MVS JCL**. This value is the DD name (including the DD:) in the *MVS JCL* (for example, DD:VSAMDATA).

The following is an example of the JCL that is required to be added to your ***MVS JCL*** that runs the Engine. **VSAMSTAN** is the name you enter into the **Temporary VSAM file in MVS JCL** text box.
P390A.STAN.VSAMDATA is the location of the temporary VSAM file on disk.
SPACE=(TRK,(30,10)) is dependent on the size of your data file.
LRECL=8192 is dependent on the size of your data record.

```
//VSAMSTAN DD DSN=P390A.STAN.VSAMDATA,DISP=(NEW,DELETE)
//            RECORG=ES,SPACE=(TRK,(30,10)),KEYLEN=7,
//            KEYOFF=2,LRECL=8192
```

More about Keyed VSAM

The **Keyed VSAM** option enables the data file to be a pre-existing VSAM KSDS file on MVS mainframe production systems.

A value is required to be entered into **File to be used in production** text box in the **Production Data Source** tab. This value is the DD name (including the DD:) in the *MVS JCL* (for example, DD:BRANCH). The **Keyed VSAM** option should only be chosen if the package file is going to run on MVS.

The following is an example of the JCL that is required to be added to your ***MVS, JCL*** that runs the Engine. **BRANCH** is the name you enter into the **File to be used in production** text box in the **Production Data Source** tab.

P390A.STAN. MVSKSDS is the location of your preexisting VSAM file on disk.
KEYLEN is the length of the key in the VSAM file.

KEYOFF is the offset of the key in the VSAM file.

LRECL is the length of a record in the VSAM file.

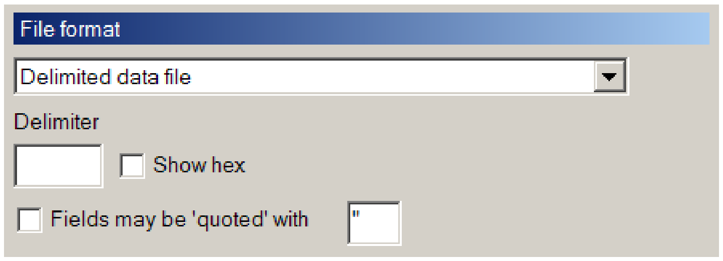
SPACE=(TRK,(30,10)) is dependent on the size of your VSAM file.

```
// BRANCH       DD DSN=P390A.STAN. MVSKSDS,DISP=SHR,
//       RECORG=KS,SPACE=(TRK,(30,10)),KEYLEN=3,
//       KEYOFF=1,LRECL=8192
```

File Format Area: Delimited Options

Delimited data source files mark the beginning and end of data by one or more delimiting characters. When you select **Delimited Data File**, more properties appear in the **File Format** area.

File Format Area – Delimited Options

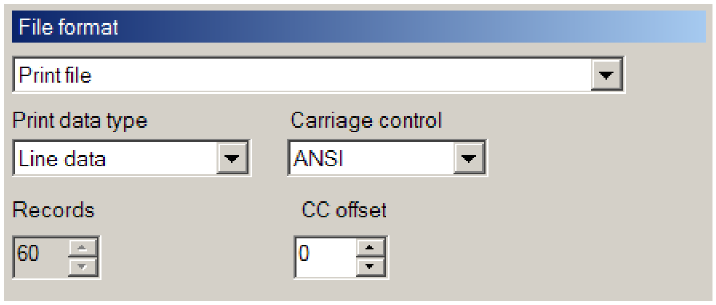


- In the **Delimiter** text box, type the delimiting character or characters used in your data source file. Very often, these characters were inserted automatically by the program that created the external data file.
- You have the option of using hexadecimal equivalents. Select the **Show hex** check box and the delimiter text box displays the hex equivalent.
- If your delimited file contains quotes around data areas:
    - Select the Fields may be ‘quoted’ with the character check box.
    - Enter the character or characters used as quotes in the open text box.

File Format Area: Print File Options

The properties that appear when you select **Print File** are unique to the **Print Miner** Module.

File Format Area – Print File


File format

Print file

Print data type: Line data

Carriage control: ANSI

Records: 60

CC offset: 0

How to identify customers in the Data File

This area appears in the second column for file types of Customer Driver File and Reference File.

The top drop-down list identifies the types of records that can be included for each customer or set. In Dialogue, files can contain any number of records for each customer. Select one of the following choices from the drop-down list to tell the Engine how to find the beginning of each set.

Each Customer Has Same Number of Records

Choose **Each customer has the same number of records** to indicate that each file contains a constant number of records. This defines the record set by the number of records. In this layout, there are no record types, and each customer must be formatted in the same way, even if there is no information in part of the record.

- Enter the **Number of Records Per Customer** in the text box below this field.

A new customer occurs on specified record type(s)

- Choose **A new customer occurs on specified record type(s**) to indicate that the data source file uses *record type indicators*.
- Specify the **Character offset of the first record-type indicator (0-based)**. This text box is 0-based for Columnar and Print files and 1-based for Delimited. Thus, if the record indicator appears in the second column, use **1**.
- Specify the **Size of the indicator**. Type the number of characters of your record type indicator.
- Sometimes record types are broken into several non-contiguous record locations. You can map this data by selecting the **Multiple Indicators** check box. When you first select the check box or press the button, an **Edit Record Indicators** dialog box appears.

You can add, delete, change the length, and change the starting position for the record indicators as they appear in the data source file.

A customer ID is on every record

- Select **A customer ID is on every record** to indicate that each record has a unique customer identifier. When the customer ID changes, Dialogue assumes that a new set of customer records is starting. In this case, all records must be of the same basic format.
- Type the **Location of the (first) record-type indicator**. This field is 0-based. Thus, if the record indicator appears in the first column, type **0**.
- Type the **Size of the indicator (Use 0 if delimited and want entire field)** by specifying the number of characters of the customer ID.
- You can select the **Multiple Indicators** check box if you want to use indicators that occur in split locations. You use the **Editor Record Indicators** dialog box as you do with the previous choice.

How Data is Identified in the Data File

This area appears in the second column for file types of **Initialization** and **Post-Sort Initialization File**. Since you use Initialization files to send any type of data to the Engine before customer processing, the data cannot be customer-dependent.

Each Record in the File has Separate Data Mappings

- Choose **Each record in the file has separate data mappings** to specify
  that each time the Engine encounters a new record (line) in the data source file, the mapping starts over.

File has Specified Record Type(s)

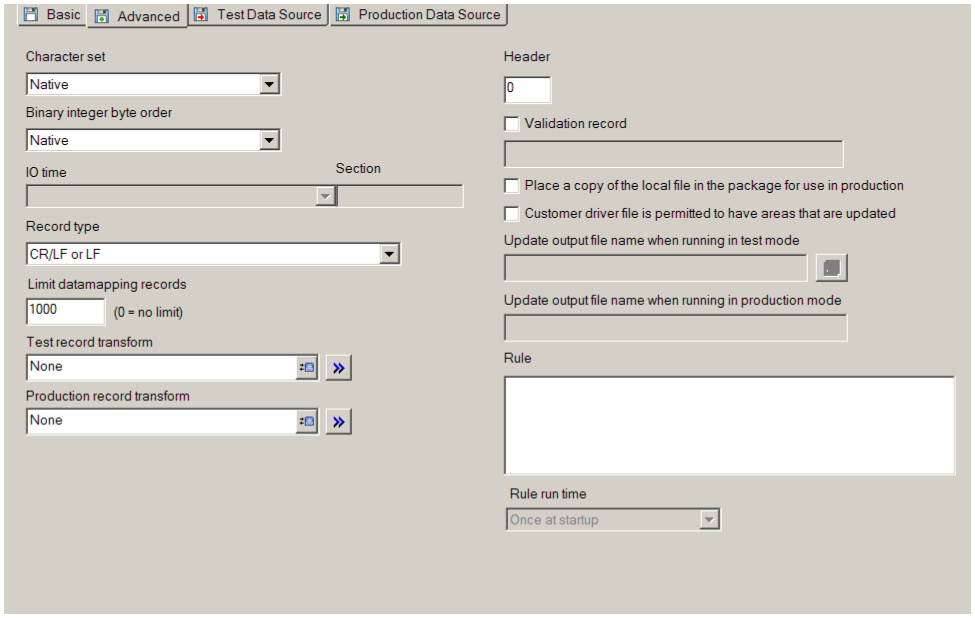
- Choose **File has specified record type(s)** to specify that each record type indicator starts a new record or section of data.
- Type the **Location of the (first) record-type indicator**. This field is 0-based. Thus, if the record indicator appears in the first column, type **0**.
- Type the **Size of the indicator (Use 0 if delimited and want entire field)** by specifying the number of characters in the indicator.

Each Record in the File has the Same Mapping

- Choose **Each record in the file has the same mapping** to indicate that all lines will have the same mapping in the same order.

4.4 Advanced Tab

Advanced Tab – Customer Driver File



Character set: Native
Binary integer byte order: Native
IO time
Section
Record type: CR/LF or LF
Limit datamapping records: 1000 (0 = no limit)
Test record transform: None
Production record transform: None
Header: 0
Validation record
Place a copy of the local file in the package for use in production
Customer driver file is permitted to have areas that are updated
Update output file name when running in test mode
Update output file name when running in production mode
Rule
Rule run time: Once at startup

Character Set

The **Character Set** option identifies for the Engine whether the data source file is stored in ASCII or EBCDIC (IBM mainframe format). Select one of the following from the **Character Set** drop-down list:

- **Native** if the data file uses the native set of the device on which you run the Engine. If you run the Engine on the mainframe, the data source file will be expected to be in EBCDIC; otherwise, it will be expected to be in ASCII. This is the default setting.
- **ASCII** if the data source file is always in ASCII, regardless of where you run the Engine.
- **EBCDIC** if the data source file is always in EBCDIC, regardless of where you run the Engine.

Binary Integer Byte Order

The **Binary integer byte order** drop-down list enables you to specify the type so the file can be read and mapped in the design environment, no matter the original platform. You have three options:

- Native
- Big-endian

- Little-endian

Note that the first time you run existing EBCDIC data files in Design Manager, they are automatically converted to big-endian.

For more information, see the *Data Files* guide.

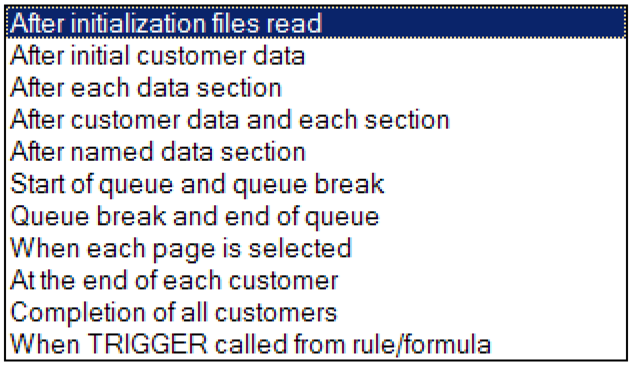
IO Time and Section

The **IO Time** option enables you to specify *when* the Engine reads or writes data. This property only pertains to data files with a **File Type** of **Reference File**, **Report File**, or **Post-Sort Report File**. The IO time drop-down list and Section box are inactive when creating a Customer driver file.

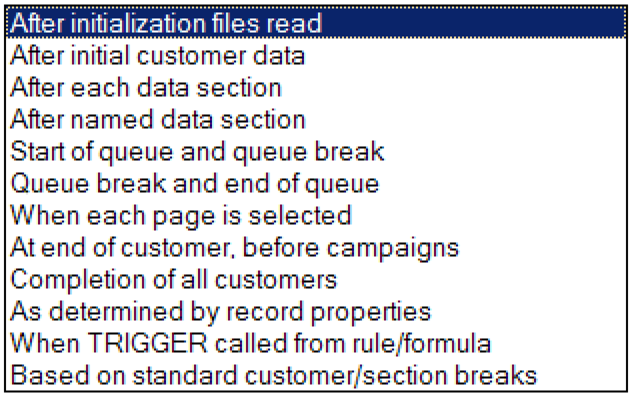
You set IO times for:

- **Reference** files – you specify a time when the Engine will *read* data records in the data source file. Dialogue computes variables based on the **IO Time** and then by rows in the data source file.
- Both types of **Report** files – you specify a time to identify when the Engine will *write* data records.

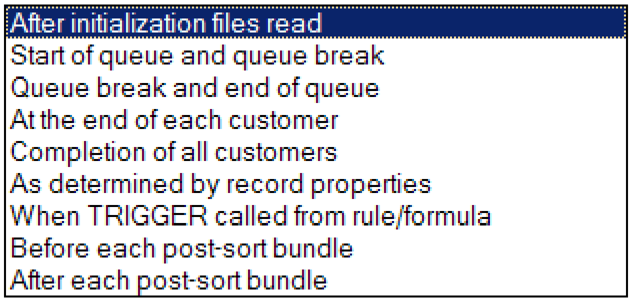
IO Time Options – Reference Files


After initialization files read
After initial customer data
After each data section
After customer data and each section
After named data section
Start of queue and queue break
Queue break and end of queue
When each page is selected
At the end of each customer
Completion of all customers
When TRIGGER called from rule/formula

IO Time Options – Report Files


After initialization files read
After initial customer data
After each data section
After named data section
Start of queue and queue break
Queue break and end of queue
When each page is selected
At end of customer, before campaigns
Completion of all customers
As determined by record properties
When TRIGGER called from rule/formula
Based on standard customer/section breaks

IO Time Options – Post-Sort Report File



After initialization files read
Start of queue and queue break
Queue break and end of queue
At the end of each customer
Completion of all customers
As determined by record properties
When TRIGGER called from rule/formula
Before each post-sort bundle
After each post-sort bundle

To summarize the options:

- **After initialization files read** – processed after the Engine reads the entire Initialization File. This is an option for Post-Sort Report, Print, Reference, and Report Files only.
- **After initial customer data** – read or written immediately after the customer's basic information. This is the default selection.
- **After customer data and each section** – read or written after the customer data is read so that the customer data can be included on a report, then section data for section summaries.
- **After each data section** – read or written after each section of data is read. This is useful for section summaries for chart/table data and reports.
- **After named data section** – read or written after a specified section of data is read. You type the name of the section in the adjacent text box. This option offers flexibility for section headers in reports.
- With this option the **Section** text box becomes available. You must identify the section by name. It is essential you use the exact same spelling for this section throughout your application.
- **Start of queue and queue break** - The Engine writes data records when the queue opens and at the top of each break file.
- **Queue break and end of queue –** read or written after each break specified in Dialogue Design Manager and after each queue is complete.
- **When each page is selected –** read or written after each page is chosen to go in a document.
- **At the end of each customer –** read or written after each customer is complete. This timing is valid for Post-Sort Report Files.
- **At end of customer, before campaigns –** read or written after each customer is complete. It is read before campaigns, if you use campaigns. The data is read just before any campaign rules are processed.

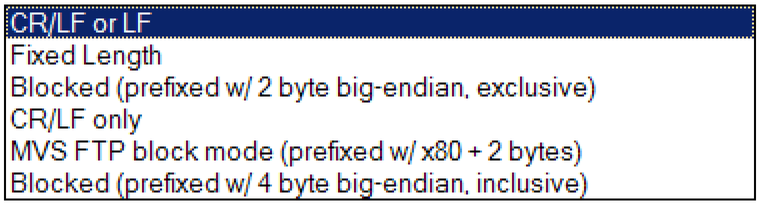
- **Completion of all customers –** read or written after the run is finished. This will create a single output record upon completion of processing.
- **As determined by Record Properties –** read or written as determined by individual properties of mapped records.
- **When TRIGGER called from rule/formula –** The Trigger function enables you to specify a specific timing in a rule or formula.
- **Based on standard Customer/Section breaks –** this is used for Report Files, and writes the standard report data based on the customer and section breaks found in the data file.
- **At end of Customer, Post-Sort –** This option is used with the **Output Sorting and Bundling** module only.

Example

For example, if you select a time of **When each page is selected**, then one record will be written for each page generated by Dialogue. You should be careful in selecting the time to ensure that all of the variables to be written to the file will be properly computed, depending on their processing time.

Record Type

Record Type Options


CR/LF or LF
Fixed Length
Blocked (prefixed w/ 2 byte big-endian, exclusive)
CR/LF only
MVS FTP block mode (prefixed w/ x80 + 2 bytes)
Blocked (prefixed w/ 4 byte big-endian, inclusive)

This option specifies the way that records begin and end in your data source file.

- **CR/LF or LF –** specifies that a carriage return and line feed or simply a line feed separates one record from the next. When possible, use this default setting. Using just a line feed separates the records by enabling one to be displayed or printed on a new line.
- **Fixed Length –** specifies that all records are the same length. You must enter the record **Length** when you select this option.

NOTE: With the MVS platform you specify the exact length. With a Windows platform you include the CR/LF in the length calculation.

- **Blocked (Prefixed, w/2-byte big-endian, exclusive) –** specifies that all records must adhere to this formatting. All records are preceded by a two-byte big-endian exclusive record length, which indicates the length of the record. This is used in the MVS environment.

- **CR/LF only –** specifies that when a carriage return or line feed occurs, the end of a line is reached, and a new record begins. The next word displays or prints as the first word in a new line.

NOTE: CR enables the next word to display or print as the first word on a new line. LF enables the record to display or print on a new line.

- **MVS FTP Block Mode (prefixed w/ x80 + 2 bytes) –** automatically reblocks for FTP in your MVS environment.
- **Blocked (Prefixed, w/4-byte big-endian, inclusive) –** specifies that all records must adhere to this formatting. All records are preceded by a four-byte, big-endian inclusive record length that is four bytes: two bytes in length and two bytes null. This is used in the MVS environment.

Limit Datamapping Records

This text box enables you to limit the number of records you view in the Edit Panel. This may be simply as a matter of visual preference, or this limit may be required to prevent memory depletion on your workstation if you view a very large file in the Edit Panel.

The default value is 1000. If you set it to 0, there will be no constraints.

Test Record Transform and Production Record Transform

You can assign a connector to a data file that points to a routine that transforms incoming data before it reaches the Engine. The routine can manipulate the data on a record-for-record basis or in a buffer (for tasks such as changing the record order).

Header Records

If the data source file uses identifying headers, type the number of records that the headers occupy in the file in the **Header** text box. When you run the Engine, this setting tells Dialogue that it does not have to produce documents for the header records, including the **Validation Record**. The records set as headers will *not* generate empty customer records.

Header Records as Initialization Information

With header records, you can set initial data in the Customer driver file without having the Engine access a separate Initialization file. You specify the number of header records and then map them with record type indicators.

Ignore in Record Properties

If you select the **Ignore** check box on the **Record Properties** dialog box, the header records are not processed. This enables you to put comments and header records at the beginning of the data file, without causing any confusion in the way the file is mapped.

Reset Time

Any variables that are mapped only in the header records should have the **Reset Time** property set to **Never Reset**.

Unknown Number of Headers

In processing, when the Engine finds a record that starts a new customer, it automatically ends the header. If you are uncertain about how many header records there are, just set the number of header records to any number that exceeds the maximum number of headers that you might have. The first customer will start normally.

Validation Record

It is not required, but it is a good practice to write a data record at the beginning of a data source file. Dialogue can then compare this *validation record* with what you enter in the **Validation Record** text box to verify the Engine is reading the correct data source file.

- Select the **Validation Record** check box.
- Enter the value of the record in the text box.

If you select the **Validation Record** check box:

- As Dialogue opens a data source file – when data mapping, performing tests, or running the Engine – it reads the first 16 characters and compares them to the first 16 characters in the **Validation Record** text box. If the characters are the same, the Engine accepts the file.
- Any characters after the 16th one are ignored.

Place a Copy of the Local File in the Package for use in Production

If you select this check box, Dialogue places a copy of this file in the package file. You may want to do this if you overwrite this file periodically, and you want to keep the current version of the file with the package. This is desirable if you reuse the package file – or if you need to retain the information for archival or reporting reasons.

If the check box is not selected, Dialogue will only read and process the necessary information from the data file. This will result in a smaller file size.

Reference Files in Processing

If the data file is a Reference file, and this box is cleared, only the information necessary for the application will be included, resulting in efficient processing. You may have many Reference files, and including all of them in every package would be undesirable.

Customer Driver File is Permitted to have Areas that are Updated

These final properties pertain to a Customer driver file only. If the check box is cleared, you cannot add updated information to this file. This setting prevents information from being overwritten in the data source file accidentally. If the check box is selected, you can update areas in your customer file if information for the customer changes during Dialogue processing. Files with update areas are updated as the Engine processes the file.

The two last text boxes on the **Advanced** tab become available when the previous option has been selected for a Customer driver file.

Update Output File Name when Running in Test Mode

A copy of the updated Customer driver file will be written to the file you specify here when running in test mode. Type the name of a local file, or use the button to locate a file.

Update Output File Name when Running in Production Mode

The Engine writes the latest data to the file you specify here when running in production mode. Specify the entire path.

Rule

Rules can be used with data files to include or exclude additional data files from being read during an Engine run. Since the Engine uses data files on an as-needed basis, this method of calling data files can decrease processing time.

The following types of data files can have rules:

- Customer driver
- Reference
- Initialization
- Post-sort initialization
- Report
- Post-sort report

For more information, see the *Data Files* guide.

When you create rules using the **Rule** dialog box, you can enter up to 250 conditions for a rule. A single rule must contain either all **and** or all **or** conditions to use this method.

For more information see the *Rules, Formulas, and Functions* guide.

Rule Run Time

The Rule run time drop-down list is inactive while defining a Customer driver file or Initialization file. When a rule is specified, it is accessed once before the file is opened. If the data file is excluded by the rule, then it is not opened or added to the internal file lists. An informational message is issued.

4.5 Data Source Tabs

Type

By default, the **Type** drop-down list has the **File** option selected.

To use this data file with the Dynamic Data Access (DDA) feature, you select **Connector** in the **Type** drop-down list. The properties in the **Data Source** tabs change.

DDA options with **Connector** will be discussed later in this class. The following sections discuss text boxes that appear with **File** in the **Type** drop-down list.

File to Use Text Boxes

On these tabs, you identify the output file names for test and production modes.

File to Use For Data Mapping

In the **File to use for data mapping box**, specify the external file used to provide information to the data file for testing purposes. Enter the file name or browse to it by clicking the 💾 button.

You can use your default text editor to create or edit a data file by clicking the 🔍 button. The specified data file opens in your default text editor. When you finish editing, save the file. The changes appear the next time you open the data file in the Edit Panel.

File to Use in Production

In the File to use in production box, specify the external data source used to provide information to the data file for actual production runs.

NOTE: If you are running the Production Engine in Windows, you can use the same path and file as the file specified on the Test Data Source tab. However, if you are running the Production Engine on another platform, such as MVS, the file name must be valid for that particular platform. Be sure these file names differ from the name of any layout file.

You can use the same file in test and production modes. The difference is in the file format necessary in Windows for test mode and the format required for your output platform. For example, on MVS production systems, the file format required might be DD:TRANSOUT or any other filename that conforms to the mainframe naming conventions.

Symbolic File Names

For the purposes of this class, you will use Windows naming conventions. If you will produce output for various platforms, you can save time if you use a *symbolic file name* rather than an actual file name. This enables you to use the same package file for multiple platforms.

For example, you can type EXOUTPUT into the **File to use in production** text box and package the application (but do not run the Engine). You can use this package file when you run the Engine in multiple production platforms. You simply specify in the *control file* for each run the actual file names for the symbolic EXOUTPUT file name, with the –FILEMAP Engine switch.

Using the EXOUTPUT example, the formats for –FILEMAP for various platforms would appear in the separate control files as:

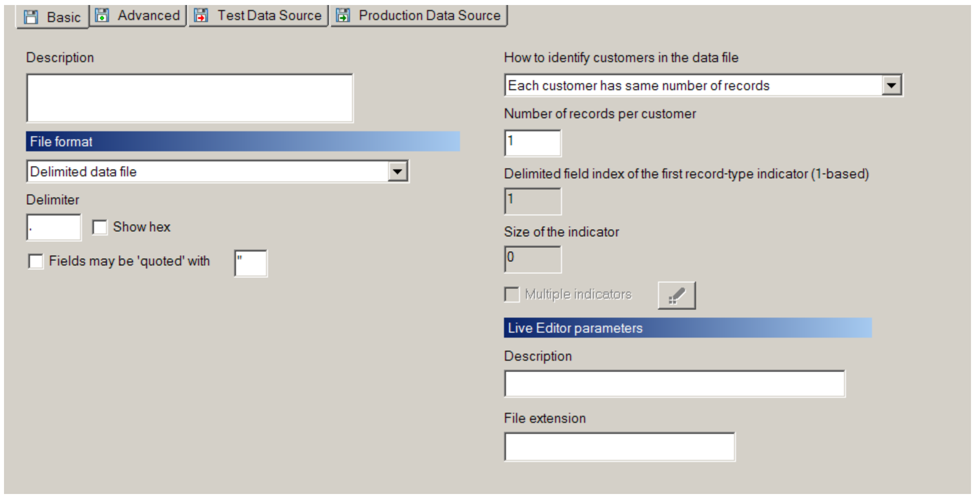
- **Windows**
  -FILEMAP=EXOUTPUT,c:\test\filename
- **UNIX**
  -FILEMAP=EXOUTPUT,/filename
- **MVS**
  -FILEMAP=EXOUTPUT,DD:EXOUT

4.6 Exercise: Define Data File Properties

A. Customer Driver File

1. Expand the **Data Files** heading under the **Exercise 02-05 Data Files** folder.
2. Drag the **Customer Driver File** to the Property Panel.
3. Click the Basic tab.
4. Type a comma as the **Delimiter**.
5. Set the **How to identify customers in the data file** property to: **Each customer has same number of records**.
6. Set **Number of records per customer** property to: **1**.

Customer Driver - Basic Tab



Description

File format: Delimited data file

Delimiter: ,

☐ Show hex

☐ Fields may be 'quoted' with "

How to identify customers in the data file: Each customer has same number of records

Number of records per customer: 1

Delimited field index of the first record-type indicator (1-based): 1

Size of the indicator: 0

☐ Multiple indicators

Description

File extension

7. Click the Test Data Source tab.
8. Set the Type property to the default of File.
9. In the File to use for data mapping property, type (or browse to):
   `C:\210 Advanced Data Concepts\Data Files\Customer Info.dat`
10. Click the Production Data Source tab.
11. In the File to use in Production property, enter: EXdriver
12. Save and close the Property Panel.

B. Initialization File

1. Drag the **Initialization File** to the Property Panel.
2. Click the **Basic** tab.

3. Ensure that the **How data is identified in the data file** drop-down list has the default of **Each record in the file has the same mapping**.
4. Click the **Test Data Source** tab.
5. Ensure that the **Type** drop-down list has the default of **File**.
6. In the **File to use for data mapping** property, browse to: `C:\210 Advanced Data Concepts\Data Files\Contact Numbers.dat`
7. Click the **Production Data Source** tab.
8. In the **File to use in Production** property, enter: **EXinit**
9. Save and close the Property Panel.

C. Reference File

1. Drag the **Reference File** to the Property Panel.
2. Click the **Basic** tab.
3. Ensure that the **Reference key layout** drop-down list displays the default of **Single variable**.
4. Select **CustomerState** from Exstream>Exercise 02-05 Data Files folder as the Key variable.
5. Ensure that the **Access** drop-down list has the default of **Memory**.
   (As you remember, this option provides the fastest processing time, but requires more resources than the other options.)
6. Enter a comma as the **Delimiter**.
7. Ensure that the **How to identify customers in the data file** drop-down displays the default of **Each customer has same number of records**.
8. Ensure that the **Number of records per customer** text box has the default of **1**.
9. Click the **Test Data Source** tab.
10. Ensure that the **Type** drop-down list has the default of **File**.
11. In the **File to use for data mapping** field, enter (or browse to): `C:\210 Advanced Data Concepts\Data Files\Area Managers.dat`
12. Click the **Production Data Source** tab.
13. In the **File to use in Production** property, enter: **EXref**
14. Save and close the Property Panel.

D. Report File

1. Drag the **Report File** to the Property Panel.
2. Click the **Basic** tab.
3. In the **Sample file to use to map report data files** box, enter (or browse to): `C:\210 Advanced Data Concepts\Data Files\Report Layout 1.txt`
4. Click the **Advanced** tab.
5. Select **Completion of all customers** from the **IO Time** drop-down list.
6. Click the **Test Data Source** tab.
7. Ensure that the **Type** drop-down list has the default of **File**.
8. In the **File to use for data mapping** field, enter (or browse to): `C:\210 Advanced Data Concepts` and enter `LetterReport.txt`
9. Click the **Production Data Source** tab.
10. In the **File to use in Production** property, enter: **EXreport**
11. Save and close the Property Panel.

Chapter 5: Mapping Data Files

5.1 In this Chapter

In this chapter, you will map existing variables to the data files you created in an earlier exercise.

Objectives

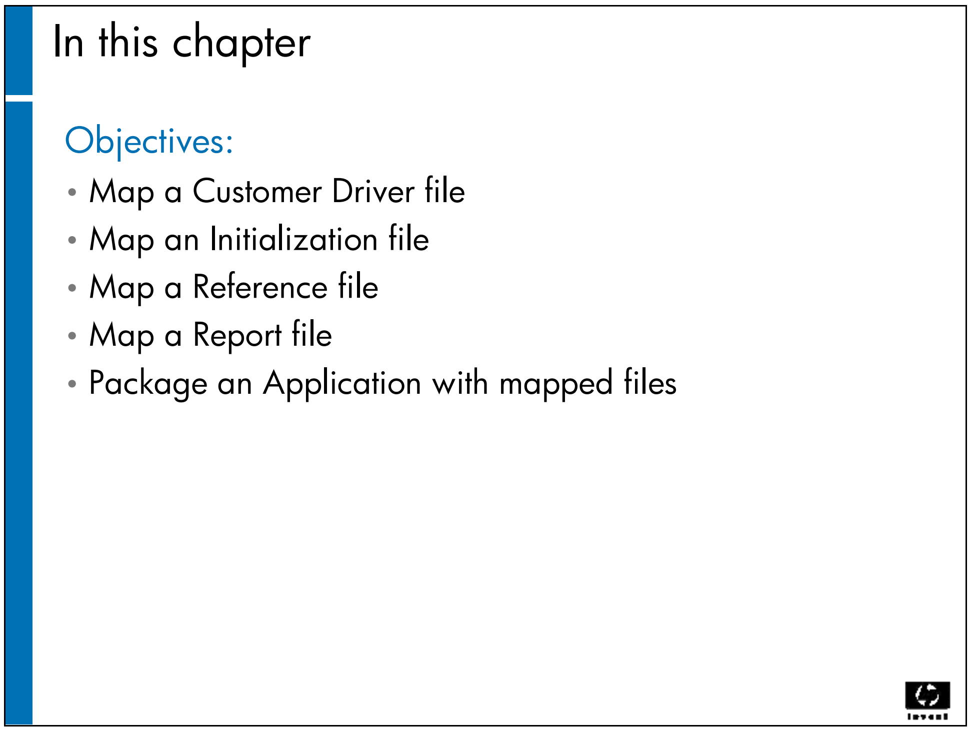
By the end of this chapter you will:

- Map a Customer Driver File.
- Map an Initialization File.
- Map a Reference File.
- Map a Report File.
- Package an application with the mapped files.

For more information

For more information, refer to the following Dialogue documentation.

■ *Data Files guide*	DataFile.pdf
■ *Variables guide*	Variable.pdf
■ *System Administration*	System.pdf

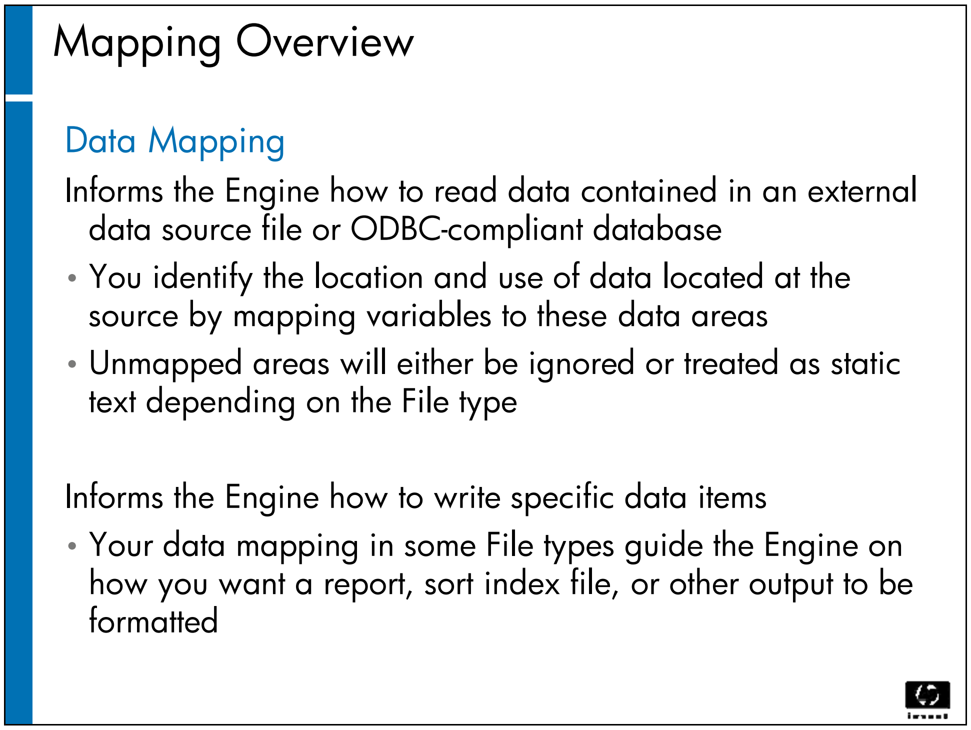


In this chapter

Objectives:

- Map a Customer Driver file
- Map an Initialization file
- Map a Reference file
- Map a Report file
- Package an Application with mapped files

5.2 Mapping Overview



Mapping Overview

Data Mapping

Informs the Engine how to read data contained in an external data source file or ODBC-compliant database

- You identify the location and use of data located at the source by mapping variables to these data areas
- Unmapped areas will either be ignored or treated as static text depending on the File type

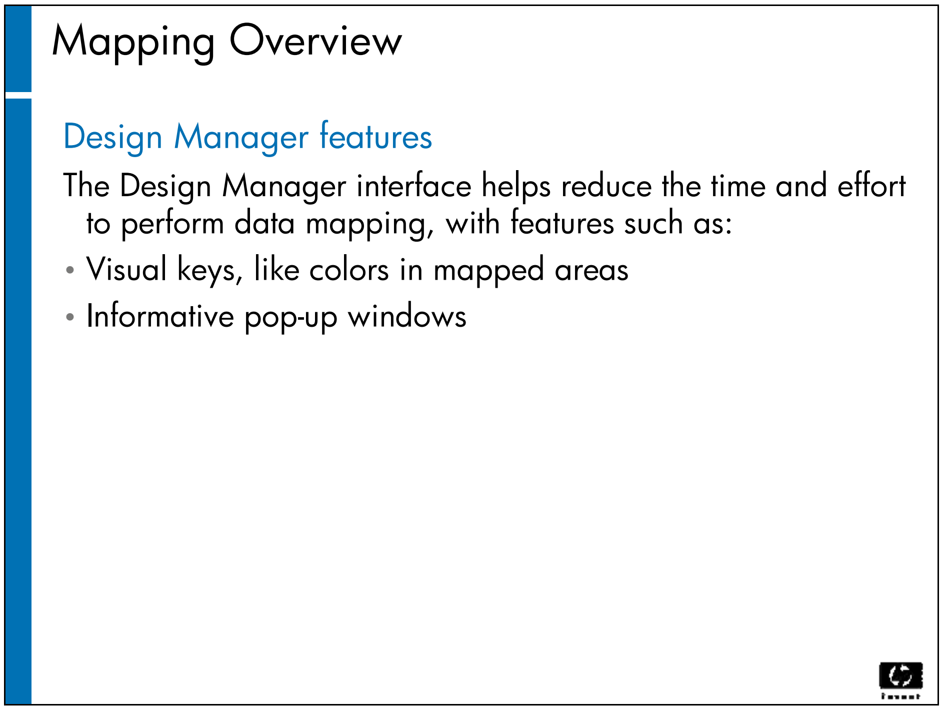
Informs the Engine how to write specific data items

- Your data mapping in some File types guide the Engine on how you want a report, sort index file, or other output to be formatted

The Engine uses the information from the data files to select files, fill text and data areas in designed objects, make inclusion rule evaluations for objects, build dynamic objects, and more.

Features that Help Speed Data Mapping

Dialogue Design Manager provides a number of time-saving features to speed the process of data mapping.



Mapping Overview

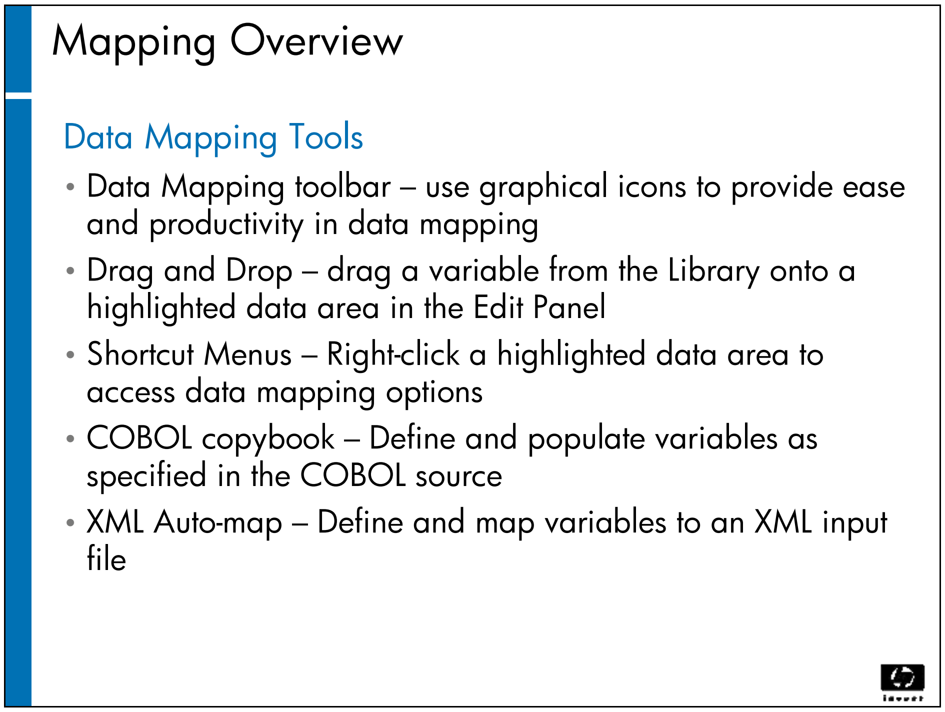
Design Manager features

The Design Manager interface helps reduce the time and effort to perform data mapping, with features such as:

- Visual keys, like colors in mapped areas
- Informative pop-up windows

- Colors, borders, and text tips visually provide helpful information with no extra keystrokes.
- When you place your pointer over a data area in the Edit Panel, a pop-up window appears. The pop-up displays the name, status, location, formatting, or other information.

Here are a number of other software tools Design Manager provides to help streamline data mapping.

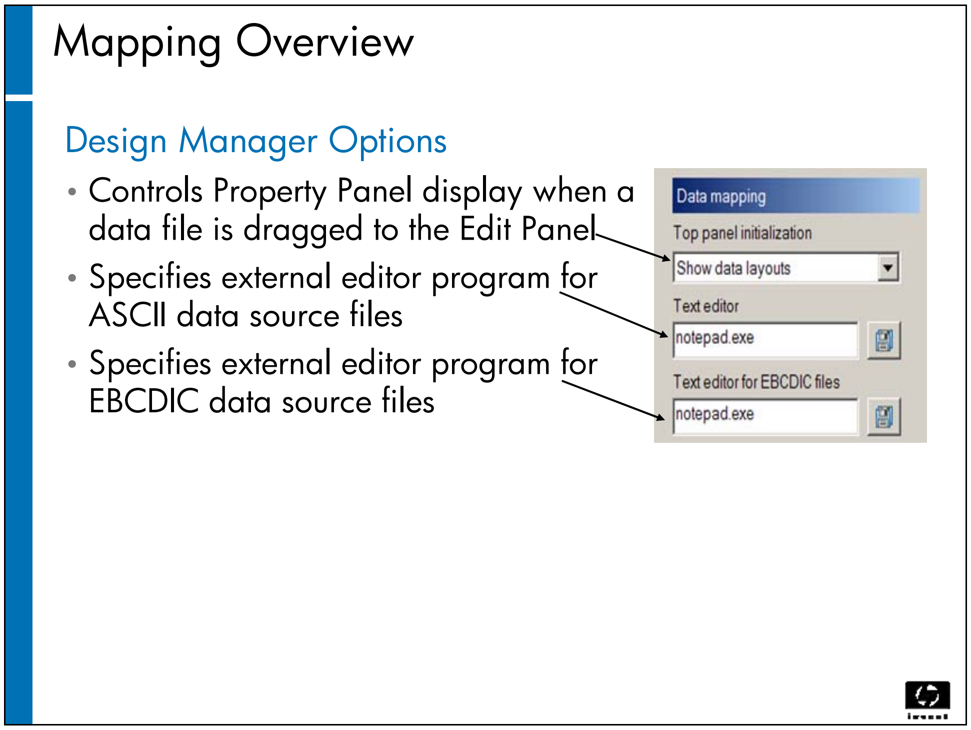


Mapping Overview

Data Mapping Tools

- Data Mapping toolbar – use graphical icons to provide ease and productivity in data mapping
- Drag and Drop – drag a variable from the Library onto a highlighted data area in the Edit Panel
- Shortcut Menus – Right-click a highlighted data area to access data mapping options
- COBOL copybook – Define and populate variables as specified in the COBOL source
- XML Auto-map – Define and map variables to an XML input file

A few other helps appear in the **Options** dialog box. To view the box, click the status bar, or select **Tools** > **Options** from the menu.



Mapping Overview

Design Manager Options

- Controls Property Panel display when a data file is dragged to the Edit Panel
- Specifies external editor program for ASCII data source files
- Specifies external editor program for EBCDIC data source files

Data mapping

Top panel initialization: Show data layouts

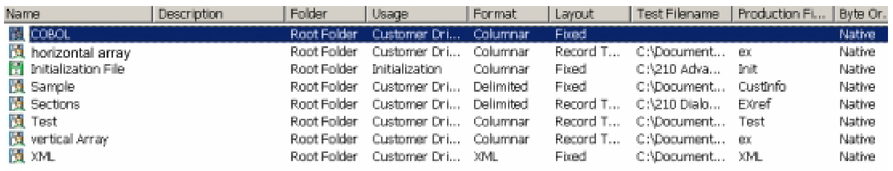
Text editor: notepad.exe

Text editor for EBCDIC files: notepad.exe

Your choice in the **Top Panel Initialization** drop-down list determines what you automatically view in the Property Panel when you drag a data file to the Edit Panel. The options are:

- **Do not show variables** – The Property Panel does not change. If you have an object open there it will remain.
- **Show data dictionary** – The Property Panel shows a list of all **Data Dictionary** variables.
- **Show data layouts** – The Property Panel automatically shows a list of all **data layouts** defined for the particular data file.

Data Layout View in Property Panel (Portion)



Name	Description	Folder	Usage	Format	Layout	Test Filename	Production Fi...	Byte Or.
COBOL		Root Folder	Customer Dri...	Columnar	Fixed			Native
horizontal array		Root Folder	Customer Dri...	Columnar	Record T...	C:\Document...	ex	Native
Initialization File		Root Folder	Initialization	Columnar	Fixed	C:\210 Adva...	Init	Native
Sample		Root Folder	Customer Dri...	Delimited	Fixed	C:\Document...	CustInfo	Native
Sections		Root Folder	Customer Dri...	Delimited	Record T...	C:\210 Dialo...	EXref	Native
Test		Root Folder	Customer Dri...	Columnar	Record T...	C:\Document...	Test	Native
vertical Array		Root Folder	Customer Dri...	Columnar	Record T...	C:\Document...	ex	Native
XML		Root Folder	Customer Dri...	XML	Fixed	C:\Document...	XML	Native

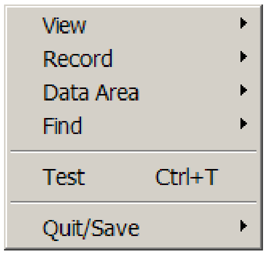
NOTE: You cannot drag and drop items into this view.

5.3 Shortcut Menu in Edit Panel

You perform data mapping in the Edit Panel. When you click a data area in the Edit Panel, a black border surrounds it, indicating that it is active.

When you right-click the Edit Panel you view a shortcut menu. When you right-click with an active data area, more options to assist you in data mapping become available.

Shortcut Menu



The topmost menu offers:

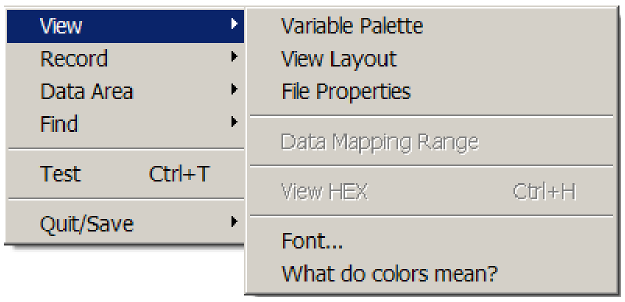
- **View** – offers options for changing the view and for viewing informational dialog boxes.
- **Record** – offers options for viewing and editing record properties.
- **Data Area** – offers options for viewing properties and mapping data areas.
- **Find** – offers options for searching data within the file.
- **Test** – enables you to test the variable mapped to a data area.
- **Quit/Save** – offers save and exit options.

The next few sections explore each of these options in detail.

5.4 Shortcut Menu: View Options

If you choose **View** from the shortcut menu, a submenu appears.

View Submenu



Variable Panel

Variable Panel opens the Variable panel.

View Layout

View Layout gives you a list of the variables used in the data file in the Property Panel. Other useful information, such as the beginning position of the variable, the format, and the length of the data area it occupies is given. Any variable that has been mapped in the file is shown. When viewing the layout for a data file, you can use the Index column for sorting. This column offers sequential numbering that can be used to return variable names to their original order after sorting by another column, such as Name.

- When you highlight a variable in this list in the Property Panel, if the variable is currently mapped in the file, it is surrounded by a black line in the Edit Panel so you can find it easily.
- When you double-click a variable in this list, it opens the **Data Area Properties** dialog box.

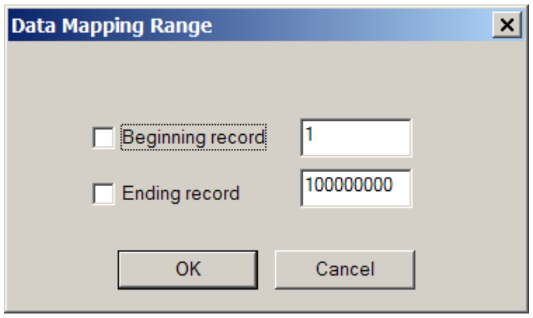
File Properties

File Properties brings up the data file in the Property Panel.

Data Mapping Range

Data Mapping Range gives you a dialog box where you can view or set the range of records to bring in Dialogue to map. Use only as many as you need for a representative sample of the data that will be mapped. That way you will not have to wait for any processing time as you test your data.

Datamapping Range

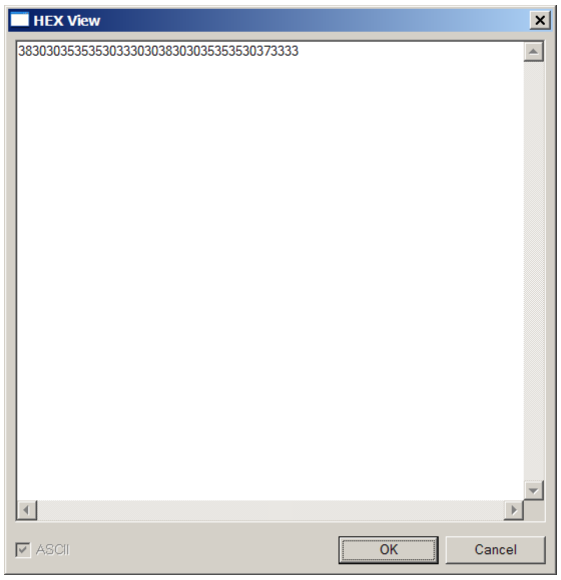


Beginning record: 1
Ending record: 100000000

View HEX

View HEX lets you see the hexadecimal value of any selected text. You can map HEX values in Dialogue data files.

View HEX

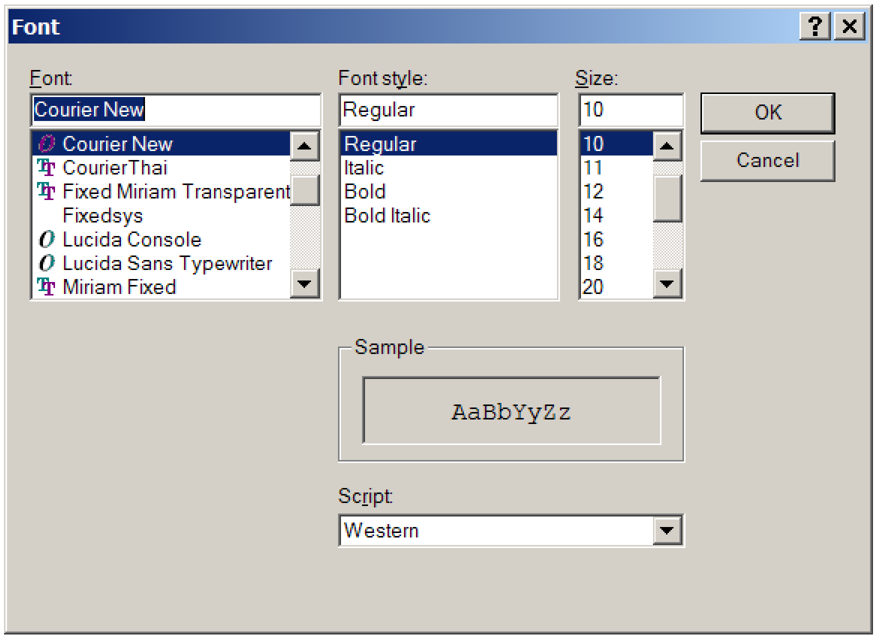


```
3830303535353033303038303035353530373333
```

Font

Font shows the current font used to display the data file, including the name, style, and size. Only fixed-space fonts appear in the list. The dialog box shows a **Sample** of the current text.

Font



If you are using the **Dynamic Data Access Module**, you can select any available **Script** on your system.

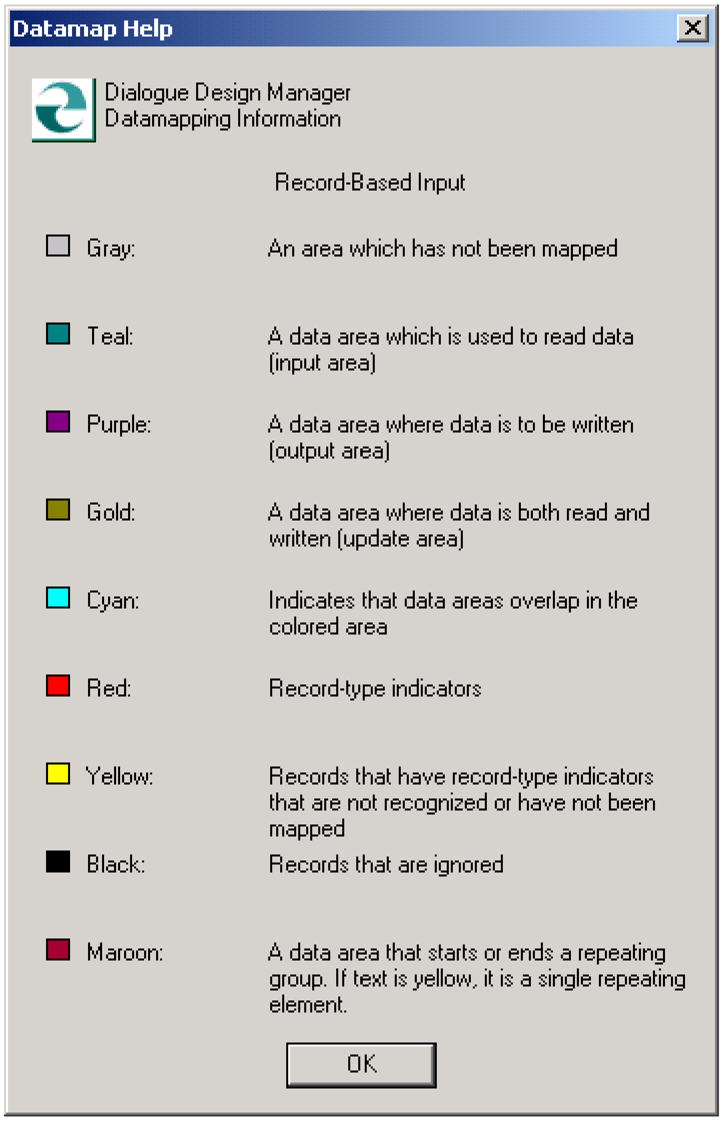
What do colors mean?

This option can be especially useful when learning data mapping.

Datamap Help

What do colors mean? shows a **Datamap Help** screen explaining the colors used to signify different types of mapped data.

Datamap Help



Dialogue Design Manager
Datamapping Information

Record-Based Input

⬜	Gray:	An area which has not been mapped
🟦	Teal:	A data area which is used to read data (input area)
🟪	Purple:	A data area where data is to be written (output area)
🟫	Gold:	A data area where data is both read and written (update area)
🟦	Cyan:	Indicates that data areas overlap in the colored area
🟥	Red:	Record-type indicators
🟨	Yellow:	Records that have record-type indicators that are not recognized or have not been mapped
⬛	Black:	Records that are ignored
🟥	Maroon:	A data area that starts or ends a repeating group. If text is yellow, it is a single repeating element.

These colors will help you quickly recognize mapped area of the data file.

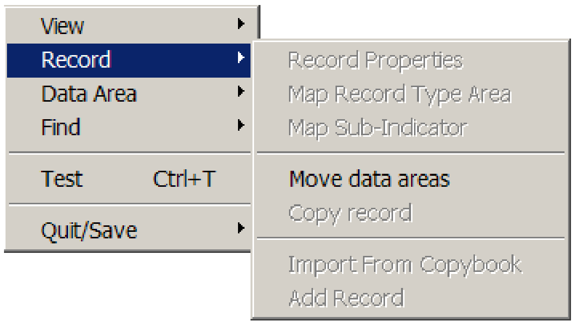
NOTE: Red denotes both indicators and sub-indicators.

NOTE: The dialog box changes based on the type of data file being edited.

5.5 Shortcut Menu: Record Options

If you choose **Record** from the shortcut menu, you see the following submenu.

Record Submenu



Record Properties

Record Properties gives you the Record Properties dialog box.

Map Record Type Area

Map Record Type Area sets the highlighted area as the area for the **Record Type Indicator**.

Map Sub-Indicator

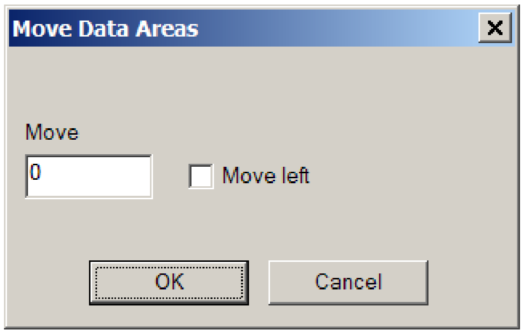
Map Sub-Indicator takes you directly to the Record Properties dialog box where you can fill in the **Sub-Indicator** area features.

Move Data Areas

Move data areas brings up a dialog box where you can move a data area *and all data areas to the right of it* – to the right or left. You will find this useful when you need to insert or delete data items in a record, or where your data is out of the boundary.

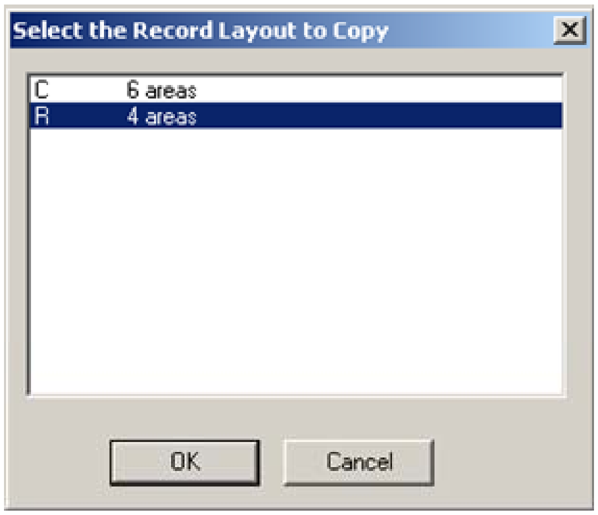
- Select the number of (single) columns (or fields if the file is delimited) to **Move**. By default, they move to the right.
- Select the **Move Left** check box if you will be moving the data items to the left, rather than to the right.

Move Data Areas Dialog Box



Copy Record

Select the Record Layout to Copy Dialog Box



C	6 areas
R	4 areas

This feature will **Copy** the existing layout from another record into the current record. The **Select the Record to Copy** dialog box will appear where you can choose an existing layout to copy. If no record layouts have been defined, as in the previous figure, you will not be able to use this function. If layouts have been defined as in the following figure, you can copy the existing layout to records with a different **record type indicator**. Notice that the dialog box tells you how many areas are defined in the record layout.

Import from Copybook

You can copy data mapping from one file to another by using a COBOL Copybook. This is discussed in depth later.

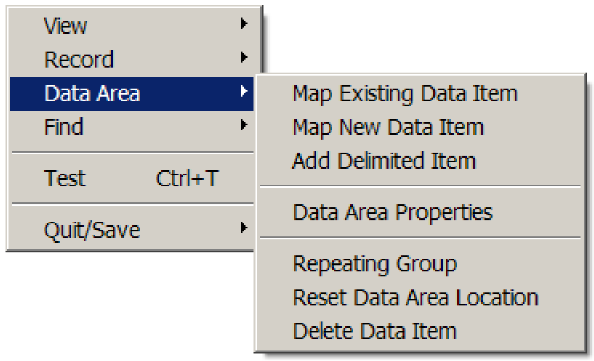
Add Record

With the **ODBC Access** Module, this option enables you to add an additional record layout to the data file.

5.6 Shortcut Menu: Data Area Options

If you choose **Data Area** from the shortcut menu, you see the following submenu.

Data Area Submenu



Map Existing Data Item lets you use an existing variable to map the selected area of the file. It gives you a list of the variables in the **Select variable** dialog box. Choose a variable to map it to the file area.

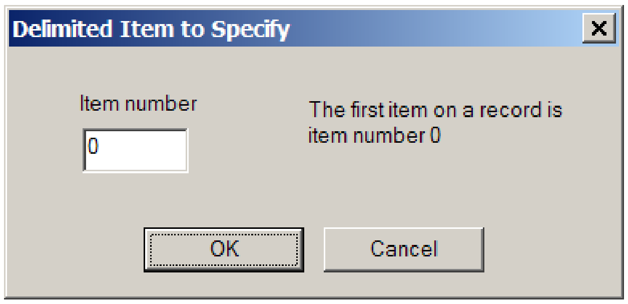
Map New Data Item lets you create a new variable to use in the data file.

Add Delimited Item lets you add a delimited data area to the file.

- First you will select a variable from the **Select the variable** dialog box.
- Then the **Delimited Item to Specify** dialog box appears where you can type the **Item Number** position of the item.

This option is useful when a delimited data area is null or contains no characters between the delimiters.

Delimited Item to Specify


Delimited Item to Specify
Item number
0
The first item on a record is item number 0
OK
Cancel

Data Area Properties opens the **Data Area Properties** dialog box so you can define other settings for your data area. You can use the **Make confidential using** drop-down list on the **Data Area Properties** dialog box to mark a data field as “private.” When you select **Design sample**, the Engine populates the data fields using the design sample.

Repeating Group lets you define a group of data areas you want the Engine to process together.

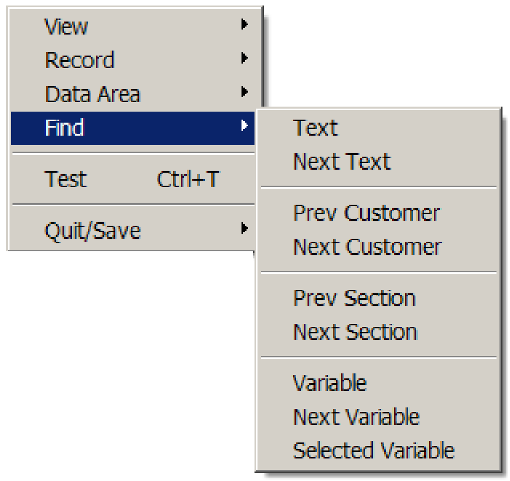
Reset Data Area Location changes the data area you have highlighted to encompass any additional or fewer spaces that you have highlighted than existed in the original data area. This way you can make the data area longer or shorter.

Delete Data Item removes the data mapping from the highlighted variable.

5.7 Shortcut Menu: Find Options

If you choose **Find** from the shortcut menu, you see the following submenu.

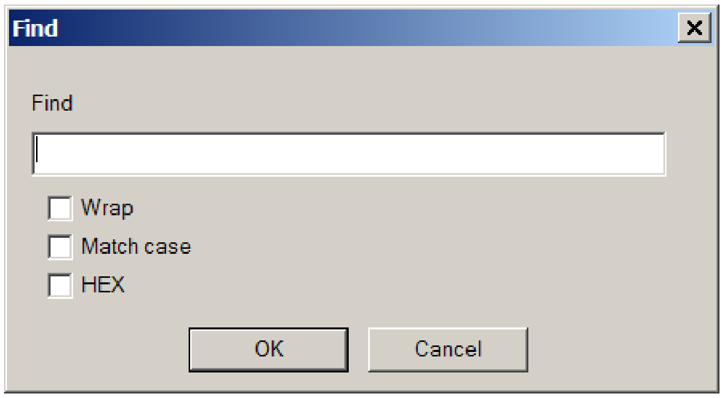
Find Submenu



Text will take you to the **Find** dialog box. Type the text you wish to find in the **Find** text box. Additionally you can check the following options to enhance the search:

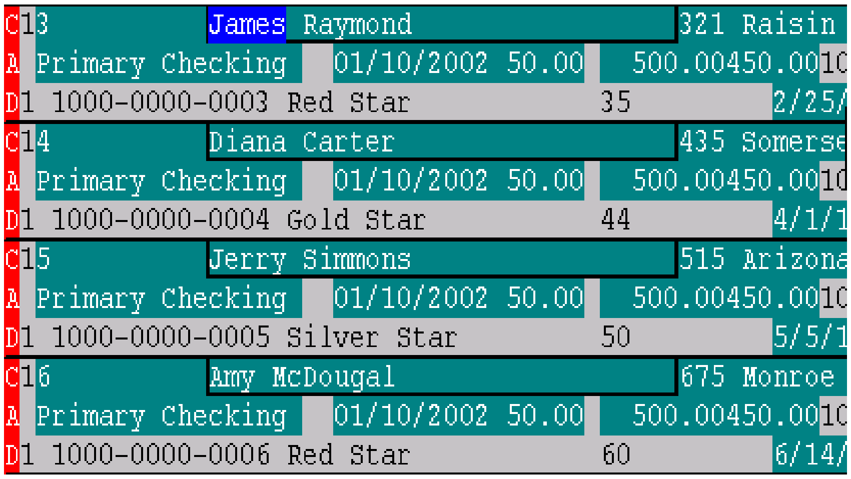
- **Wrap** – Points to the end of the file and searches the entire data file. If you clear this check box, Dialogue searches from only the current point forward.
- **Match case** - Finds only instances of text that exactly match the upper- and lower-case characteristics of your search text.
- **HEX** - Searches for the hexadecimal value that you enter in the Find box.

Find Dialog Box



After you click **OK**, Dialogue highlights the first instance of the text. If needed, Design Manager may reposition your view in the Edit Panel so you can more easily see the text.

Text Found in File



```
C13            James Raymond                     321 Raisin
A Primary Checking  01/10/2002 50.00  500.00450.0010
D1 1000-0000-0003 Red Star               35          2/25/
C14            Diana Carter                      435 Somerse
A Primary Checking  01/10/2002 50.00  500.00450.0010
D1 1000-0000-0004 Gold Star              44          4/1/1
C15            Jerry Simmons                     515 Arizona
A Primary Checking  01/10/2002 50.00  500.00450.0010
D1 1000-0000-0005 Silver Star            50          5/5/1
C16            Amy McDougal                      675 Monroe
A Primary Checking  01/10/2002 50.00  500.00450.0010
D1 1000-0000-0006 Red Star               60          6/14/
```

Next Text will take you to the next occurrence of the text you just found located any number of screens away from your current position.

Prev Customer will take you to the previous customer in the data file. This is useful when the file contains large amounts of information for each customer and the previous customer may be located any number of screens away from your current position in the data file window.

Next Customer will take you to the next customer in the data file.

Prev Section will take you to the previous section in the data file. This is useful when the file contains large amounts of sectional information for each customer and the previous customer may be located a screen away in the data file window. Remember, sections are divided by a medium weight black line.

Next Section will take you to the next section in the data file.

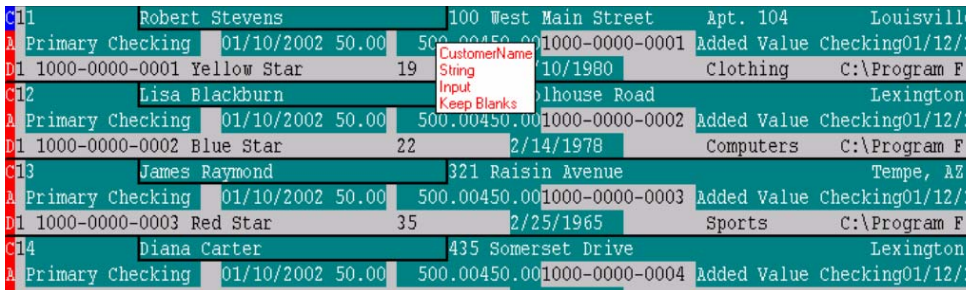
If you choose **Variable**, the **Select Variable** dialog box appears.

If you select a variable by clicking and dragging to highlight it on in the Edit Panel, and then choose **Selected Variable**, all the areas containing the selected variable will be outlined with a bold black line.

You select the variable from the list or begin typing the name of the variable. To try this, select **CustomerName** in any data file that contains it.

Dialogue will select the **CustomerName** variable throughout your data file.

Customer's Name in Data File



```
C11           Robert Stevens               100 West Main Street     Apt. 104        Louisvill
A Primary Checking  01/10/2002 50.00   50         1000-0000-0001 Added Value Checking01/12/
D1 1000-0000-0001 Yellow Star          19          10/1980          Clothing      C:\Program F
C12           Lisa Blackburn                      lhouse Road                       Lexington
A Primary Checking  01/10/2002 50.00   500.00450.001000-0000-0002 Added Value Checking01/12/
D1 1000-0000-0002 Blue Star            22         2/14/1978         Computers     C:\Program F
C13           James Raymond                321 Raisin Avenue                         Tempe, AZ
A Primary Checking  01/10/2002 50.00   500.00450.001000-0000-0003 Added Value Checking01/12/
D1 1000-0000-0003 Red Star             35         2/25/1965         Sports        C:\Program F
C14           Diana Carter                 435 Somerset Drive                       Lexington
A Primary Checking  01/10/2002 50.00   500.00450.001000-0000-0004 Added Value Checking01/12/
```

If you choose **Selected Variable**, you will scroll to the next instance of the selected variable used in the file. Again, this is very useful in files with long sections of data, and especially in delimited files where the information does not fall neatly into columns as with **columnar** files.

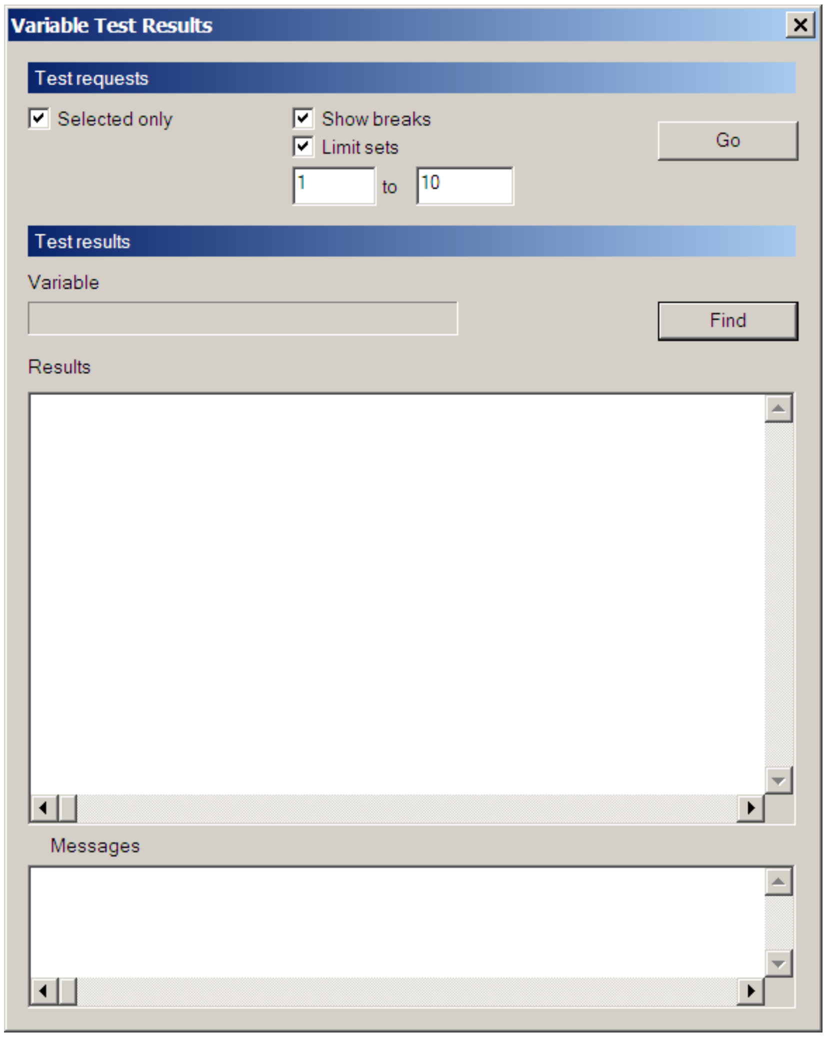
5.8 Shortcut Menu: Test Option

You can **Test** a selected variable, or all the variables, in your mapping. Testing from the Edit Panel is not supported for **XML** and **Print Miner** data.

Test Procedure

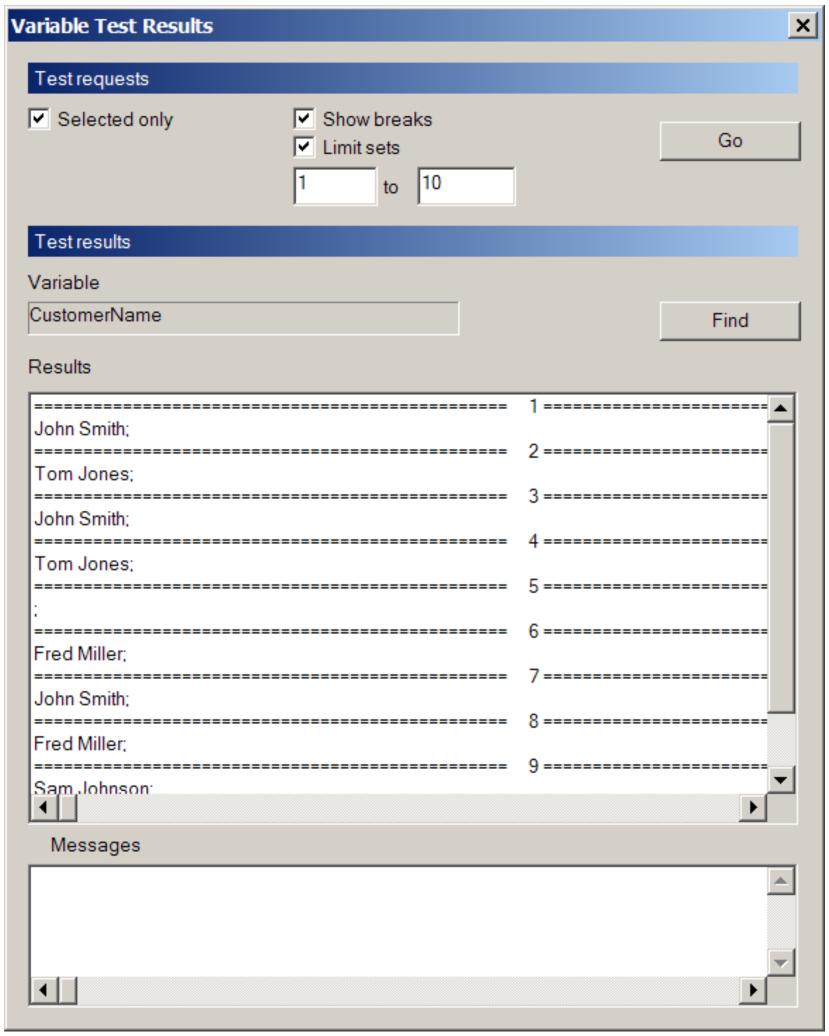
- Right-click a variable in the data file.
- Choose **Test**.
- Select the **Selected Only** check box if you want to find values for the one variable you highlighted. Clear the check box to view values for all mapped variables in this data file.
- Select the **Show Breaks** check box to see where each customer ends.
- If you want to see only a certain sample of the data file:
  - Select the **Limit Sets** check box.
  - Type a beginning and ending range of customers in the text boxes.
- Click **Go** to begin the test.

Variable Test Results Dialog Box



Selected only: ✓
Show breaks: ✓
Limit sets: ✓
1 to 10
Variable
Results
Messages

Variable Test Results – Testing All Variables



[✓] Selected only
[✓] Show breaks
[✓] Limit sets
1 to 10

Variable
CustomerName

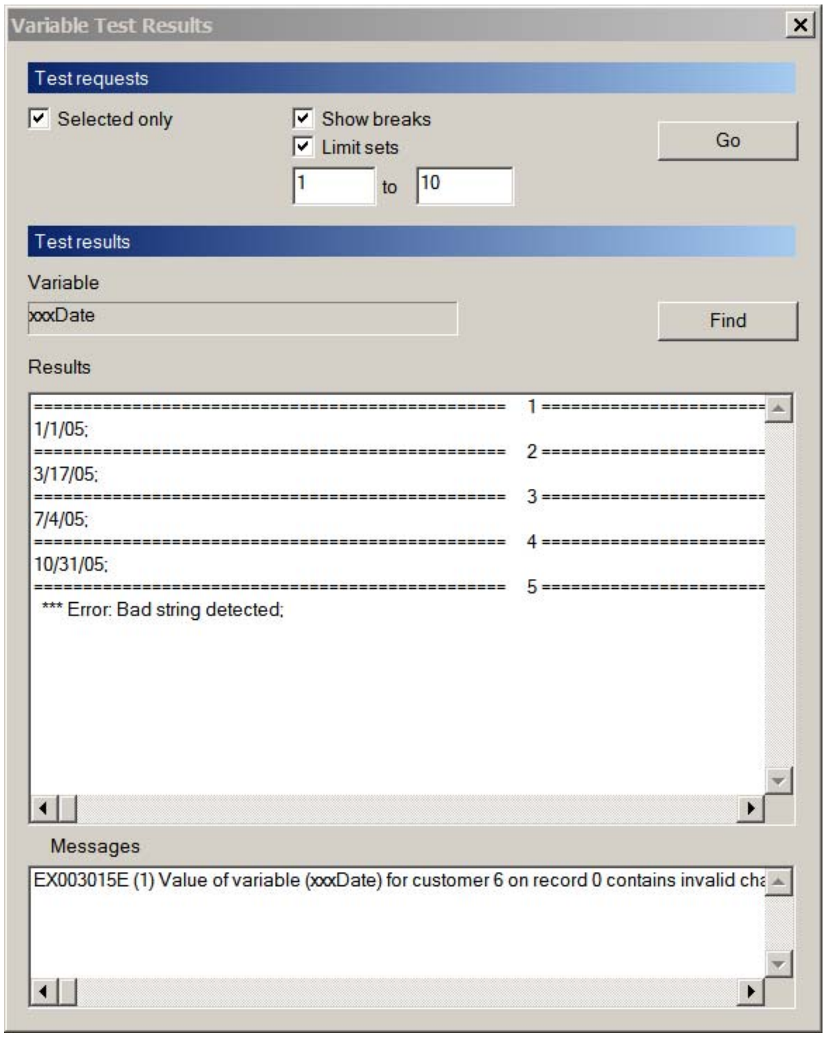
Results

```
================================================ 1 ======================
John Smith;
================================================ 2 ======================
Tom Jones;
================================================ 3 ======================
John Smith;
================================================ 4 ======================
Tom Jones;
================================================ 5 ======================
;
================================================ 6 ======================
Fred Miller;
================================================ 7 ======================
John Smith;
================================================ 8 ======================
Fred Miller;
================================================ 9 ======================
Sam Johnson;
```

Error Messages

Error messages appear in the **Messages** window at the bottom of the dialog box.

Variable Test Results – With Date Error



Selected only [✓]
Show breaks [✓]
Limit sets [✓]
1 to 10

Variable
xxxDate

Results

```
==================================================   1 ======================
1/1/05;
==================================================   2 ======================
3/17/05;
==================================================   3 ======================
7/4/05;
==================================================   4 ======================
10/31/05;
==================================================   5 ======================
 *** Error: Bad string detected;
```

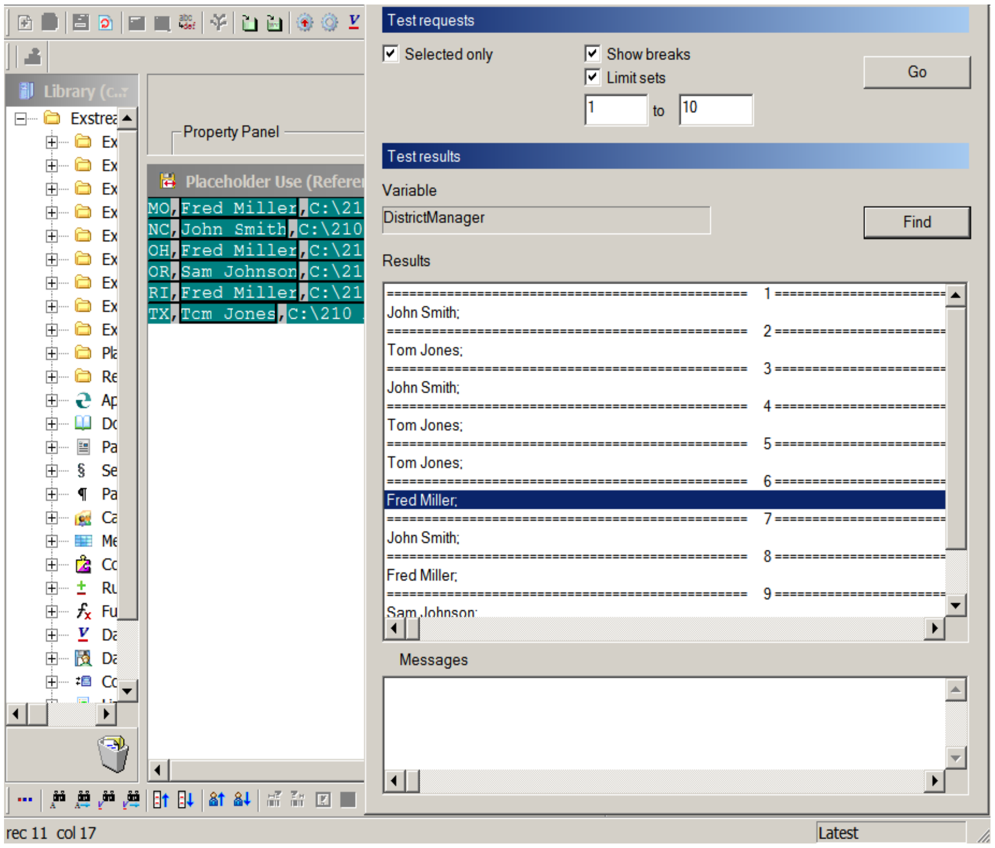
Messages

```
EX003015E (1) Value of variable (xxxDate) for customer 6 on record 0 contains invalid cha
```

Find Data Area in Edit Panel

To find a data area in the Edit Panel that corresponds to a value in the **Results** window or the **Messages** window, double-click (or highlight and click the **Find** button) an item in the **Variable Test Results** dialog box. Dialogue places the corresponding variable at the top of the variable list in the Edit Panel. For best results, arrange your windows as shown.

Test results dialog box with variable in Edit Panel



```
MO,Fred Miller,C:\21
NC,John Smith,C:\210
OH,Fred Miller,C:\21
OR,Sam Johnson,C:\21
RI,Fred Miller,C:\21
TX,Tcm Jones,C:\210
```

[✓] Selected only [✓] Show breaks [✓] Limit sets 1 to 10

Variable: DistrictManager

Results

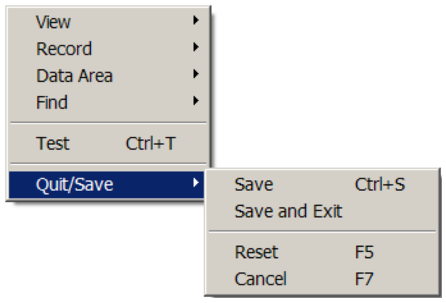
```
==================== 1 ====================
John Smith;
==================== 2 ====================
Tom Jones;
==================== 3 ====================
John Smith;
==================== 4 ====================
Tom Jones;
==================== 5 ====================
Tom Jones;
==================== 6 ====================
Fred Miller;
==================== 7 ====================
John Smith;
==================== 8 ====================
Fred Miller;
==================== 9 ====================
Sam Johnson;
```

rec 11 col 17 Latest

5.9 Shortcut Menu: Quit/Save Options

If you choose **Quit/Save** from the shortcut menu, you get a submenu.

Quit/Save Submenu



- **Save** will save the current data file in the active panel.
- **Save and Exit** will save the current data file in the active panel and close it.
- **Reset** will reset the properties on the active panel to the last saved version.
- **Cancel** will close the active panel without saving changes to the file.

5.10 Record Indicators

Simple Files

A **simple** Customer driver file or Reference file contains one set of data for each customer. The Engine reads all the data for the customer and then builds the customer document. This type of Customer driver file does not need record type indicators.

If your file has no record type indicators, then all records are assumed to have the same record format. For example, if the file properties on the **Basic** tab have been set to **Each customer has same number of records**, each record is defined as one to a line.

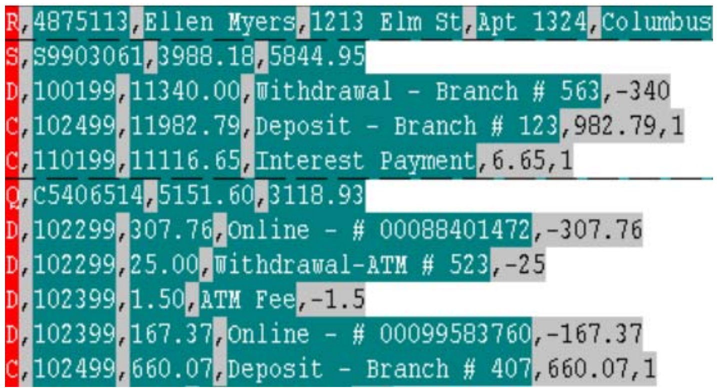
Record Type Indicators

Dialogue Design Manager uses *record type indicators* to separate logical groups of information within a customer. A record type indicator identifies the start of a section (or section within a section) in the flow of the data. Often, you see an indicator as a single character, but this is not a requirement: an indicator can be a longer string of characters up to 2 gigabytes. There must be a unique record indicator to indicate the start of each unique section of data.

Records can have multiple indicators. Each section can contain any number of records and any number of sub-sections with their own record type indicators. In the example on the next page, the following lists the characters in the first column:

- R indicates a new customer
- S indicates a savings account section
- D indicates a debit within that account
- C indicates a credit within that account
- Q indicates a checking account section for the customer with its own set of Ds and Cs (debits and credits).

Mapped File in Edit Panel

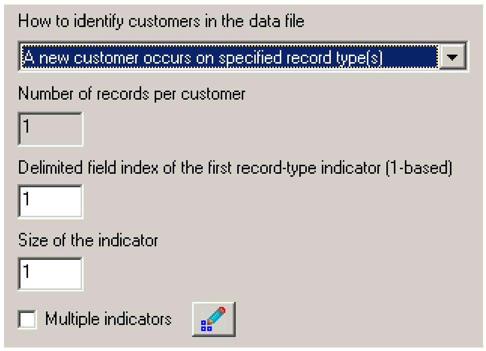


```
R,4875113,Ellen Myers,1213 Elm St,Apt 1324,Columbus
S,S9903061,3988.18,5844.95
D,100199,11340.00,Withdrawal - Branch # 563,-340
C,102499,11982.79,Deposit - Branch # 123,982.79,1
C,110199,11116.65,Interest Payment,6.65,1
Q,C5406514,5151.60,3118.93
D,102299,307.76,Online - # 00088401472,-307.76
D,102299,25.00,Withdrawal-ATM # 523,-25
D,102399,1.50,ATM Fee,-1.5
D,102399,167.37,Online - # 00099583760,-167.37
C,102499,660.07,Deposit - Branch # 407,660.07,1
```

Map a Record Type Indicator

- Select an unmapped date file object for which you want to include data sections, and drag it into the Property Panel.
- Under **How to identify customers in the data file** select **A new customer occurs on specified record type(s)**.

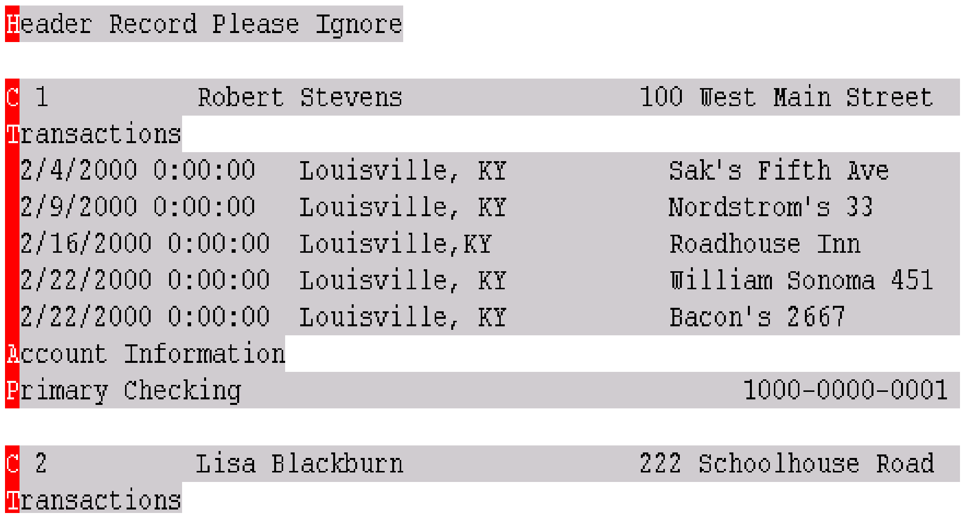
Customer Driver File Properties – Basic Tab


How to identify customers in the data file
A new customer occurs on specified record type(s)
Number of records per customer
1
Delimited field index of the first record-type indicator (1-based)
1
Size of the indicator
1
Multiple indicators

- If the indicators will appear in the far left column, set the Character offset of the (first) record-type indicator to 0 for columnar or 1 for delimited. If the indicators are not in the first column, insert the number of columns. The field is offset to the right of the first column. Column one is zero on columnar and print files and column one is one on delimited files.
- Set its **Size of the indicator** to the number of characters. In the current example on this page each indicator is **1** character long.
- When record indicators are not contiguous in the data source file, select the **Multiple Indicators** check box. The **Edit Record Indicators** dialog box appears. Show **Start** and **Length** for the indicators you **Add**. Click **OK** to return to the Property Panel.
- **Save** your data file object.

When you look at the file in the Edit Panel, the indicators will appear in red as in the following figure.

Customer Driver Data File in Edit Panel

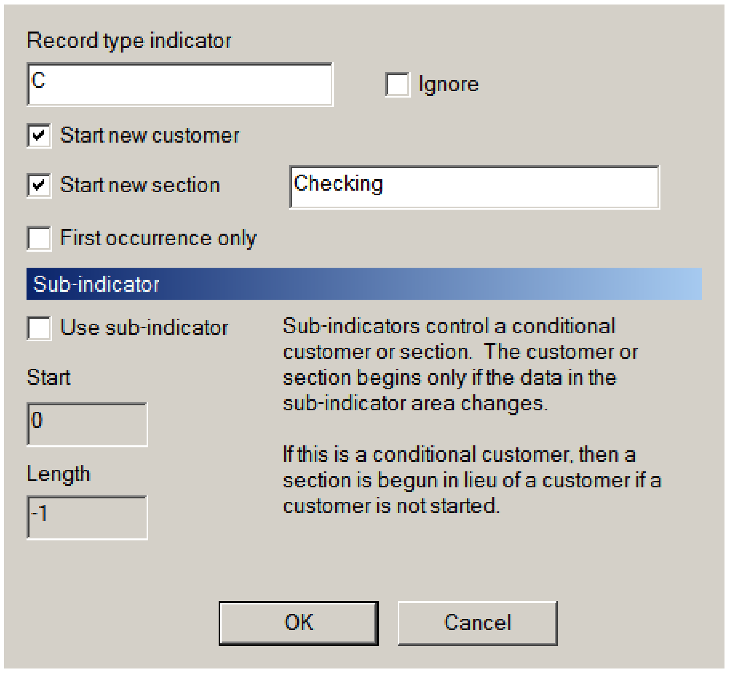


```
Header Record Please Ignore

C 1           Robert Stevens                100 West Main Street
Transactions
 2/4/2000 0:00:00   Louisville, KY          Sak's Fifth Ave
 2/9/2000 0:00:00   Louisville, KY          Nordstrom's 33
 2/16/2000 0:00:00  Louisville,KY           Roadhouse Inn
 2/22/2000 0:00:00  Louisville, KY          William Sonoma 451
 2/22/2000 0:00:00  Louisville, KY          Bacon's 2667
Account Information
Primary Checking                                  1000-0000-0001

C 2           Lisa Blackburn                222 Schoolhouse Road
Transactions
```

Start a New Customer

- Highlight the portion of the record that starts a new customer. In the previous figure, it is "C."
- Right-click to access the *shortcut menu.*
- Select the **Record** -> **Record Properties** option. The **Record Properties** dialog box appears.

Record Properties Dialog Box



Record type indicator: C

Ignore [ ]

[✓] Start new customer

[✓] Start new section: Checking

[ ] First occurrence only

Sub-indicator

[ ] Use sub-indicator

Start: 0

Length: -1

Sub-indicators control a conditional customer or section. The customer or section begins only if the data in the sub-indicator area changes.

If this is a conditional customer, then a section is begun in lieu of a customer if a customer is not started.

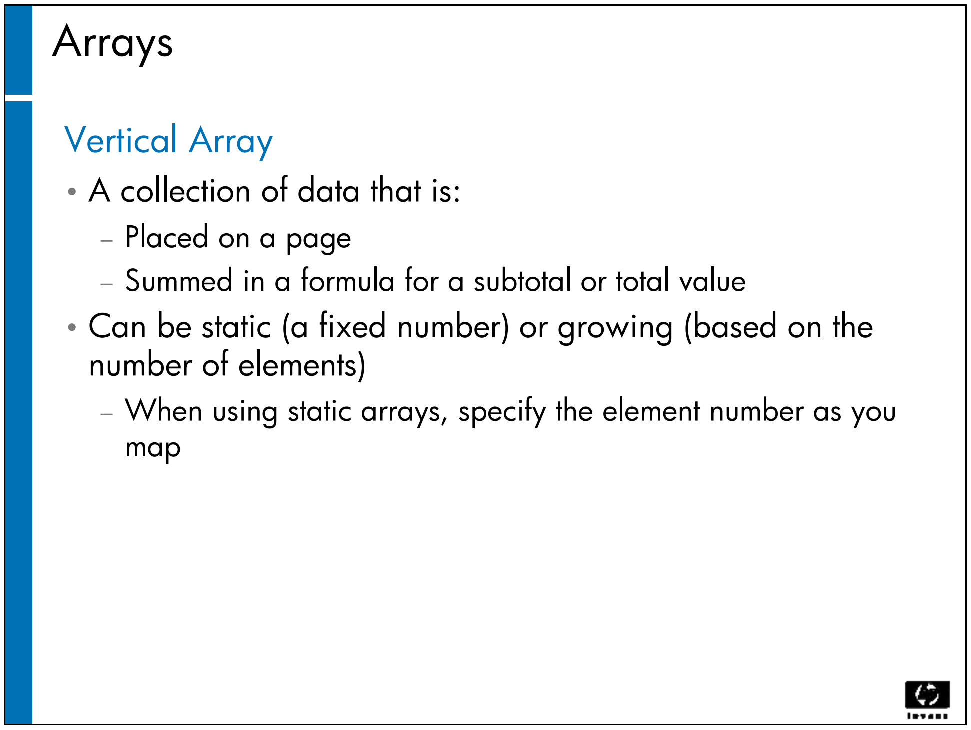
Notice that the **Record Properties** dialog box:

- Shows the character(s) used for the *record type indicator.*
- Has an **Ignore** check box where you can select to ignore this record type. You select this check box when you want certain record types to be ignored. One of these types may be header records that occur in your data. This allows mapped records to be "turned off" if the data is not needed. This tells the Engine not to process these records, to enhance performance.

- Select the **Start New Customer** check box if the record type you selected starts a new customer.
- If you are not defining section data, you are finished with this dialog box. Click **OK**. Otherwise, proceed to the next discussion.

5.11 Arrays

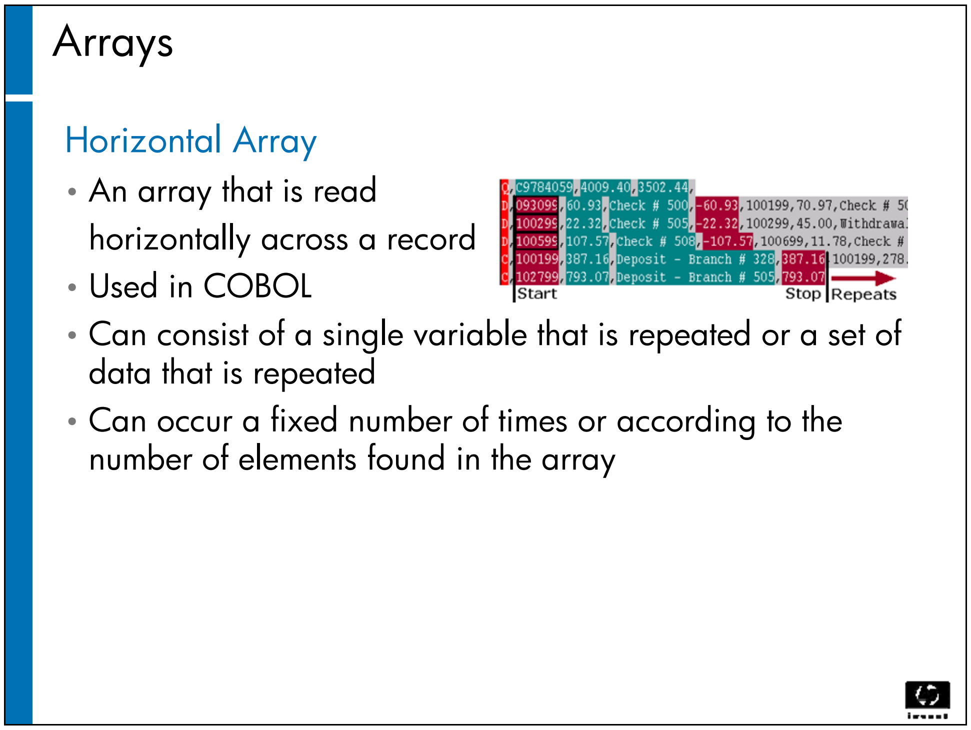
You can use vertical arrays with transaction data. They are arranged on separate lines in the data source file.



Arrays

Vertical Array

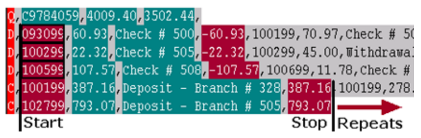
- A collection of data that is:
  - Placed on a page
  - Summed in a formula for a subtotal or total value
- Can be static (a fixed number) or growing (based on the number of elements)
  - When using static arrays, specify the element number as you map



Arrays

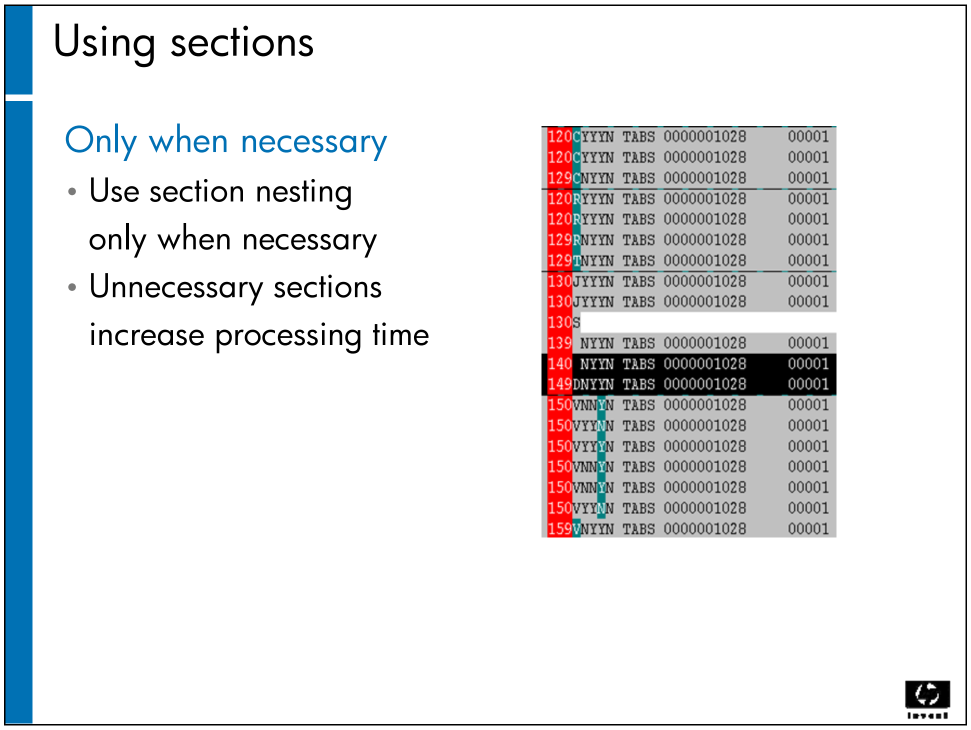
Horizontal Array

- An array that is read horizontally across a record
- Used in COBOL



```
Q,C9784059,4009.40,3502.44,
D,093099,60.93,Check # 500,-60.93,100199,70.97,Check # 5(
D,100299,22.32,Check # 505,-22.32,100299,45.00,Withdrawal
D,100599,107.57,Check # 508,-107.57,100699,11.78,Check #
C,100199,387.16,Deposit - Branch # 328,387.16,100199,278.
C,102799,793.07,Deposit - Branch # 505,793.07
```

- Can consist of a single variable that is repeated or a set of data that is repeated
- Can occur a fixed number of times or according to the number of elements found in the array



Using sections

Only when necessary

- Use section nesting only when necessary
- Unnecessary sections increase processing time

```
120CYYYN TABS 0000001028      00001
120CYYYN TABS 0000001028      00001
129CNYYN TABS 0000001028      00001
120RYYYN TABS 0000001028      00001
120RYYYN TABS 0000001028      00001
129RNYYN TABS 0000001028      00001
129TNYYN TABS 0000001028      00001
130JYYYN TABS 0000001028      00001
130JYYYN TABS 0000001028      00001
130S
139 NYYN TABS 0000001028      00001
140 NYYN TABS 0000001028      00001
149DNYYN TABS 0000001028      00001
150VNNYN TABS 0000001028      00001
150VYYNN TABS 0000001028      00001
150VYYYN TABS 0000001028      00001
150VNNYN TABS 0000001028      00001
150VNNYN TABS 0000001028      00001
150VYYNN TABS 0000001028      00001
159VNYYN TABS 0000001028      00001
```

One variable can simultaneously be horizontal and vertical. Dialogue collects data horizontally, then vertically.

5.12 Section-Based Files

Section-Based Files

A **section-based** Customer driver file or Reference file contains multiple *sections* of data for each customer. Typically, each section of the data file causes a specific document or set of data in a table to be created for the customer. For example, in an application designed for a brokerage firm, a given customer may have one section of data for each retirement account and another section of data for each non-retirement account.

A section-based data file may also contain ***section-independent*** data. This is high-level information about a customer that the Engine reads *before* any of the sections. For example, this could be address block information the Engine reads before reading sections with specific account data.

Record type indicators *are* required in this type of file to:

- Identify different sections.
- Separate section-independent data from section-dependent data.

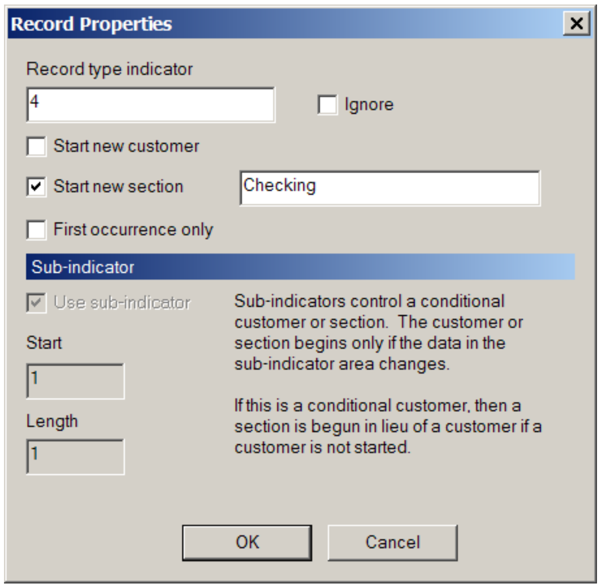
These data sections are also called **Named Sections** in some Dialogue options.

Start a New Section

To define a record as the start of a new section:

- Right-click **Record** > **Record Properties** and select the **Start New Section** check box. Additional options become available.
- Type a name for the new section in the text box next to the field. This section name must be typed *exactly* in *all* instances throughout the application.

Record Properties Dialog Box (Top Portion)

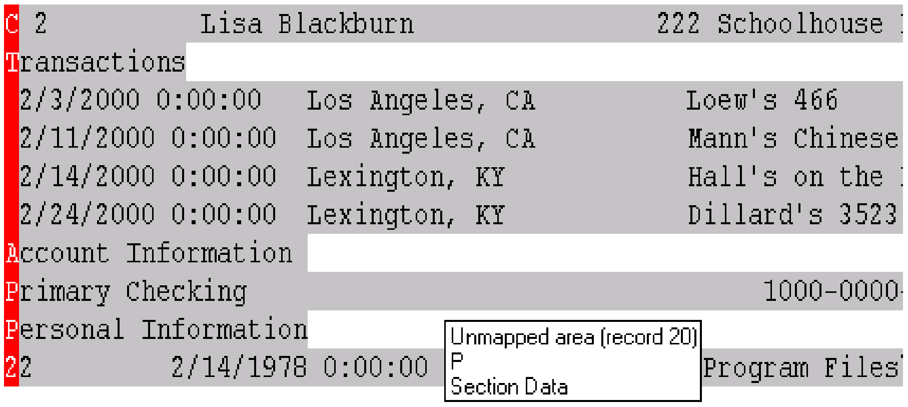


Sub-indicators control a conditional customer or section. The customer or section begins only if the data in the sub-indicator area changes.

If this is a conditional customer, then a section is begun in lieu of a customer if a customer is not started.

- Select the **First Occurrence Only** check box to have the Engine read this indicator only the first time it appears for each customer. In the figure, two records begin with "P," Primary Checking and Personal Information. If you selected the **First Occurrence Only** check box, only the first "P," Primary Checking, will be recognized as a record type indicator.

Customer File with Duplicate Indicators



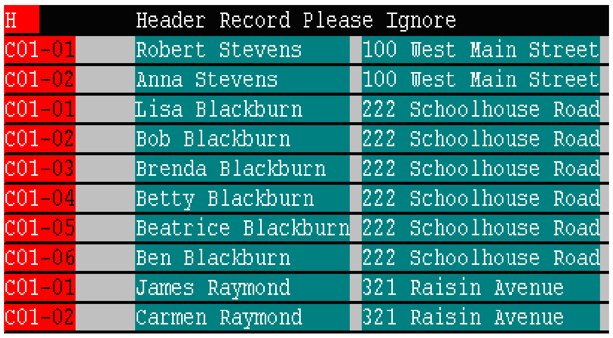
```
C 2          Lisa Blackburn             222 Schoolhouse
Transactions
 2/3/2000 0:00:00   Los Angeles, CA         Loew's 466
 2/11/2000 0:00:00  Los Angeles, CA         Mann's Chinese
 2/14/2000 0:00:00  Lexington, KY           Hall's on the
 2/24/2000 0:00:00  Lexington, KY           Dillard's 3523
Account Information
Primary Checking                                  1000-0000
Personal Information
22         2/14/1978 0:00:00                Program Files
```

Start a Subsection

To set up a section within a section, select the **Use Sub-Indicator** check box. which you can use if the indicator is a section break for either:

- A conditional customer, or
- A conditional section.

Data File with Sub-Indicators (-01, -02, and So On)



H	Header Record Please Ignore	
C01-01	Robert Stevens	100 West Main Street
C01-02	Anna Stevens	100 West Main Street
C01-01	Lisa Blackburn	222 Schoolhouse Road
C01-02	Bob Blackburn	222 Schoolhouse Road
C01-03	Brenda Blackburn	222 Schoolhouse Road
C01-04	Betty Blackburn	222 Schoolhouse Road
C01-05	Beatrice Blackburn	222 Schoolhouse Road
C01-06	Ben Blackburn	222 Schoolhouse Road
C01-01	James Raymond	321 Raisin Avenue
C01-02	Carmen Raymond	321 Raisin Avenue

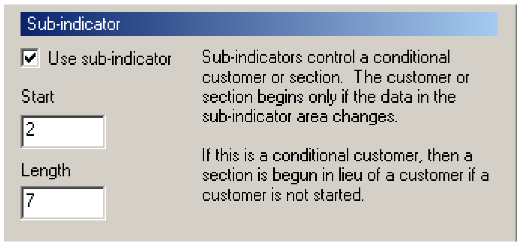
The conditional customer or section begins only if the sub-indicator changes. This can be used for subaccounts under one account, or particular transaction types for an account.

If the sub-indicator indicates a conditional customer, (you have checked **Start New Customer** for a previous subsection) but this subsection does not indicate a new customer, a section for the conditional customer begins. This is an important feature for **Householding**, when there is more than one person at an address, but all information goes in one envelope. When this indicator changes, or is blank, a new household is started.

The **Start** and **Length** text boxes can be used with both **columnar** and **delimited** data files.

- For columnar files, **Start** indicates the starting column, while **Length** refers to the length of the subindicator.
- For delimited files, **Start** refers to the field count while **Length** is set to 0 to indicate the entire field.

Record Properties dialog box (Top Portion)



Sub-indicator

☑ Use sub-indicator

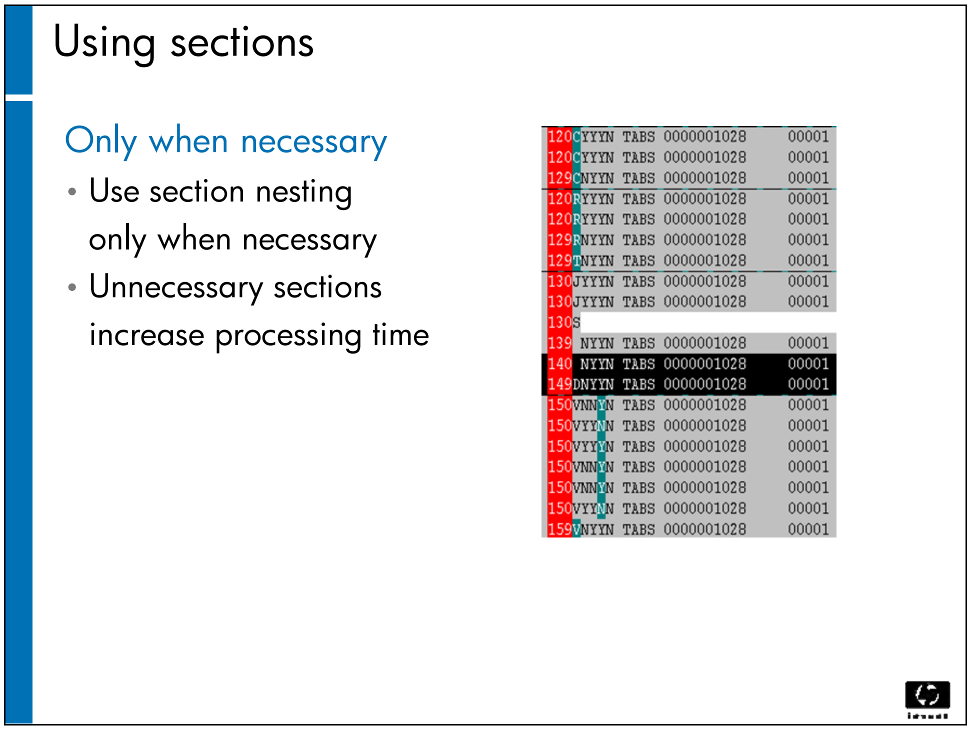
Start: 2

Length: 7

Sub-indicators control a conditional customer or section. The customer or section begins only if the data in the sub-indicator area changes.

If this is a conditional customer, then a section is begun in lieu of a customer if a customer is not started.

Using Sections

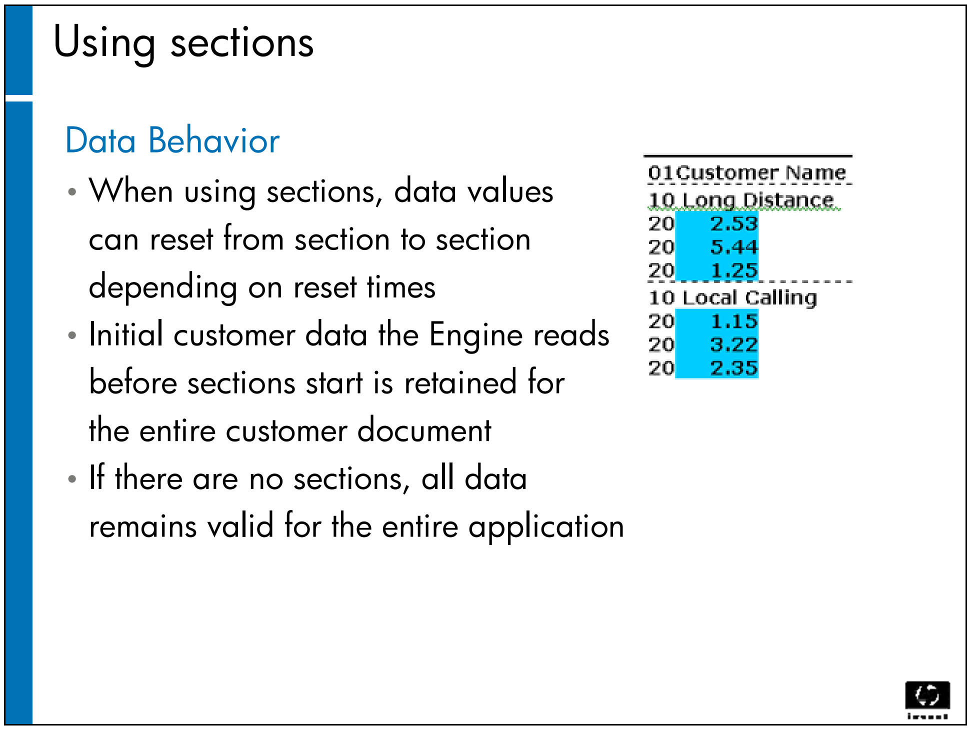


Using sections

Only when necessary

- Use section nesting only when necessary
- Unnecessary sections increase processing time

```
120CYYYN TABS 0000001028     00001
120CYYYN TABS 0000001028     00001
129CNYYN TABS 0000001028     00001
120RYYYN TABS 0000001028     00001
120RYYYN TABS 0000001028     00001
129RNYYN TABS 0000001028     00001
129TNYYN TABS 0000001028     00001
130JYYYN TABS 0000001028     00001
130JYYYN TABS 0000001028     00001
130S
139 NYYN TABS 0000001028     00001
140 NYYN TABS 0000001028     00001
149DNYYN TABS 0000001028     00001
150VNNYN TABS 0000001028     00001
150VYYNN TABS 0000001028     00001
150VYYYN TABS 0000001028     00001
150VNNYN TABS 0000001028     00001
150VNNYN TABS 0000001028     00001
150VYYNN TABS 0000001028     00001
159VNYYN TABS 0000001028     00001
```



Using sections

Data Behavior

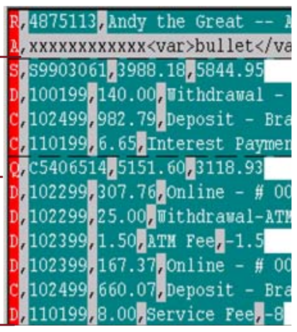
- When using sections, data values can reset from section to section depending on reset times
- Initial customer data the Engine reads before sections start is retained for the entire customer document
- If there are no sections, all data remains valid for the entire application

```
01Customer Name
10 Long Distance
20     2.53
20     5.44
20     1.25
10 Local Calling
20     1.15
20     3.22
20     2.35
```

Using sections

How the Engine processes sections

- The Dialogue Engine reads records until it finds a section or another customer
- Customer pages and rules are processed
  - Reads records until the next section or reaches end of customer
  - Processes data, rules and tables
  - All array variables are reset
    - Reads records until next Section or End of Customer
    - Processes data, rules and tables



```
R,4875113,Andy the Great --
A,xxxxxxxxxxxx<var>bullet</va
S,S9903061,3988.18,5844.95
D,100199,140.00,Withdrawal -
C,102499,982.79,Deposit - Bra
C,110199,6.65,Interest Paymen
Q,C5406514,5151.60,3118.93
D,102299,307.76,Online - # 00
D,102299,25.00,Withdrawal-ATM
D,102399,1.50,ATM Fee,-1.5
D,102399,167.37,Online - # 00
C,102499,660.07,Deposit - Bra
D,110199,8.00,Service Fee,-8
```

At end of customer, the Engine finishes processing the pages and sends them to the next production stage

Using sections

Table processing

1) The Engine reads all records until it reaches the first section
2) The Engine begins filling tables with customer data, according to sections
3) Tables are designed as skeletons. As the Engine processes each section, the tables grow
4) At the end of the customer, Dialogue breaks the tables into page-sized segments, processes barcodes, and places messages into available space

5.13 Design Tips for Efficient Data Handling

Data File Structure

- Data files must have an identifiable structure, such as a record ID or customer ID on every record.
- Arrange your data in the order the Engine will read it.
- Columnar files are easier to read data and see sections than delimited files.
- Delimited files can be faster to map than columnar files.
- Use Reference files for information that is frequently updated. If more than one variable points to the Reference file, use an Auxiliary layout file.
- Initialization files are useful for information that can be processed at the beginning of the run and is static for all customers throughout the application.
- Initialization files are a good place for using string lookups.

Number of Data Files

- Use as few files as possible.
- Use one Customer driver file, rather than several, if possible.
- It is more efficient to use a single Customer driver file than adding a Reference file to the application. Likewise, it is more efficient to use one Reference file rather than several Reference files.
- When files become too large for your equipment (like memory or processor speed) to handle, split them. Place static information in one Customer driver file.

Section Data

- Use section data only when necessary. Sections consume processing time.
- Arrange your sections in the order the Engine will read them.
- A single record cannot start both a section and a customer.
- Section data can be timed independently from initial information.
- Variable names, such as “account” can be reused in different sections, yet will process together in groups according to record properties.

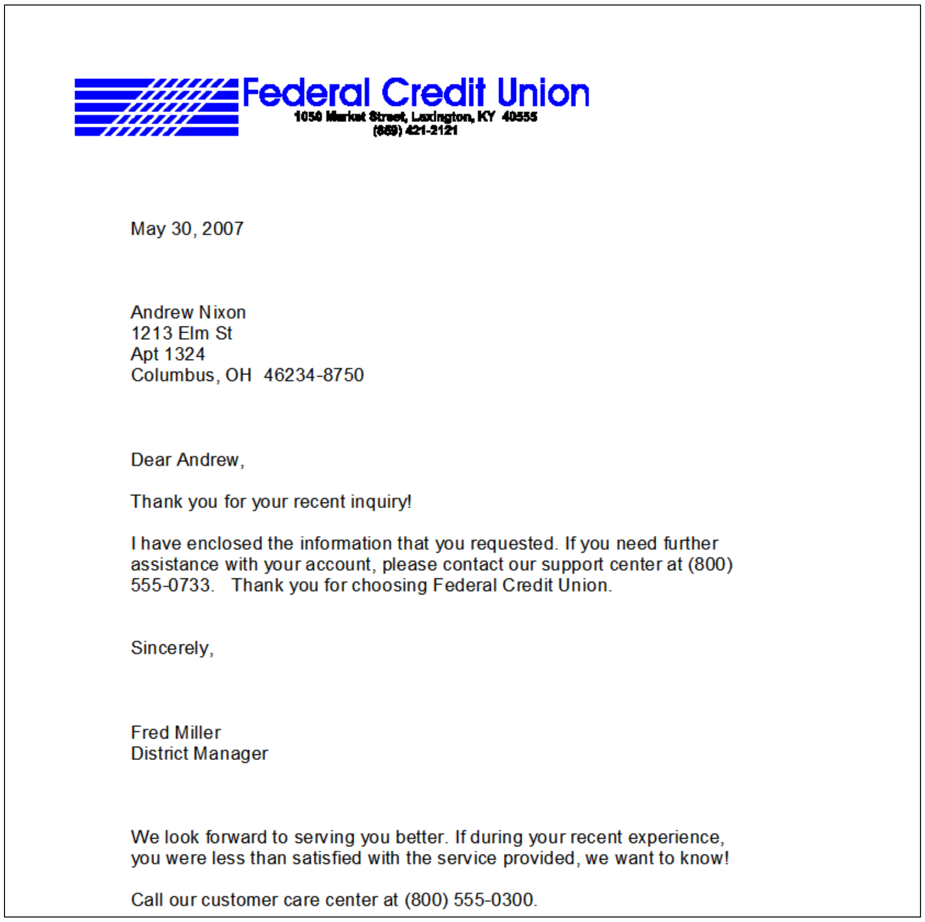
Messages

- Messages can exist in a Reference file, Initialization file, or Customer driver file. Use the method that enables you to access the messages at the proper time. If the messages are the same for all customers, you can use an Initialization file. If the message names differ for each customer, or constantly change, you can use a Reference file.
- You can use DynaMessages to bring in real-time messages. DynaMessages are messages that are loaded from an external file at Engine run-time. They are controlled by an XML input file that specifies the message to be inserted at Engine run-time. The benefit of DynaMessages is that you do not have to repackage in order to use rapidly changing RTF text messages when your actual production runs involve thousands or millions of customers.
- Placeholder messages and documents can be updated until run time, since the file name remains the same, while contents can change.

5.14 Exercise: Map Data Files

In this exercise you will map variables to the data files you defined in an earlier exercise. Then you will test your mapping by composing a letter.

Composed Letter Page



Federal Credit Union
1050 Market Street, Lexington, KY 40555
(859) 421-2121

May 30, 2007

Andrew Nixon
1213 Elm St
Apt 1324
Columbus, OH 46234-8750

Dear Andrew,

Thank you for your recent inquiry!

I have enclosed the information that you requested. If you need further assistance with your account, please contact our support center at (800) 555-0733. Thank you for choosing Federal Credit Union.

Sincerely,

Fred Miller
District Manager

We look forward to serving you better. If during your recent experience, you were less than satisfied with the service provided, we want to know!

Call our customer care center at (800) 555-0300.

A. Map a Customer Driver File

1. In Design Manager, navigate to: **Exercise 02-05 Data Files**.
2. Drag the **Customer Driver File** to the Edit Panel.
3. Open the **Variable Panel**.
4. Ensure the **Variable Panel** is filtered to the Exercise 02-05 Data Files folder.

Map each data area as described in the following sections.

Map First Data Area

5. Double click the first data area.
6. Double-click the **CustomerAccount** variable from the **Variable Panel**. The **Data Area Properties** dialog box appears. No changes are needed.
7. Click **OK**.

This mapping appears in a teal color, indicating the Engine will read this area as input data.

Map the Remaining Areas

8. Repeat the procedure above to map the remaining data areas. No changes are needed to **Data Area Properties**.

Data Area	Variable to map
Second data area	**CustomerFname**
Third data area	**CustomerLname**
Fourth data area	**CustomerAddress1**
Fifth data area	**CustomerAddress2**
Sixth data area	**CustomerCity**
Seventh data area	**CustomerState**
Eighth data area	**CustomerZip**
Ninth data area	**CustomerSSN**

9. Save and close the Edit Panel.

B. Map an Initialization File

1. Drag the **Initialization File** to the Edit Panel.
2. Using the steps from mapping the Customer driver file, map the two data areas in this file.

Data Area	Variable to map	Start	Length
First data area	**CustomerServiceNum**	**1**	**10**
Second data area	**CustomerCareNum**	**11**	**10**

3. Save and close the Edit Panel.

C. Map a Reference File

1. Drag the **Reference File** to the Edit Panel.
2. Using the steps from mapping the Customer driver file, map the two data areas in this file.

Data Area	Variable to map
First data area	**CustomerState**
Second data area	**DistrictManager**

3. Save and close the Edit Panel.

D. Map a Report File

1. Drag the **Report File** to the Edit Panel.
2. Ensure the Variable Panel is filtered to the Exstream folder.
3. Using the steps from mapping the Customer driver file, map the three data areas in this file.

Data Area	Variable to map	Start	Length
RunDate	**SYS_DateCurrent**	**26**	**25**
TotalCustomers	**SYS_CustomerInRun**	**26**	**15**
TotalPages	**SYS_PageTotalInRun**	**26**	**15**

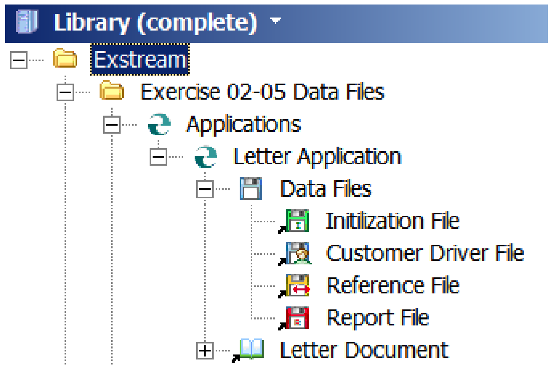
4. Save and close the Edit Panel.
5. Close the **Variable Panel**.

NOTE: You will receive a message stating there are unmapped records that will be written to the report as-is.

E. Add the Data Files to an Application

1. Expand the Exercise 02-05 Data Files folder.
2. Expand the Applications heading.
3. Drag the Initialization File to the Letter Application.
4. Drag the Customer driver File to the Letter Application.
5. Drag the Reference File to the Letter Application.
6. Drag the Report File to the Letter Application.
7. Ensure that the data files appear in the order shown in the following figure.

Data Files in Correct Order



Library (complete)
Exstream
Exercise 02-05 Data Files
Applications
Letter Application
Data Files
Initilization File
Customer Driver File
Reference File
Report File
Letter Document

Data files can be moved, copied, referenced, or cloned to the applications that require the information. In the Library, data files must appear in this order under the application: Initialization files, Customer driver files, Reference files, then Report files.

F. Add the Mapped Variables to a Page

1. Expand the Exercise 02-05 Letter Document folder.
2. Drag the Letter Page to the Edit panel.
   Variables names are typed on the page in red.
3. Use the Variable panel to replace the typed variable names with actual variables from the Exstream 02-05 folder and the Exstream folder.
4. Change text color to black after replacing variables.
5. Save the page and exit Designer.

G. Package and Run the Engine

1. Expand the Exercise 02-05 Applications folder.
2. Package the Letter Application and review results in the Viewer.
3. Using Windows Explorer, browse to C:\210 Advanced Data Concepts\Data Files\Letter Report.txt.
4. Open Letter Report.txt and review the result.

Chapter 6: COBOL Copybook

6.1 In this Chapter

Dialogue supports a COBOL Copybook feature that significantly reduces the time to create variables and map data areas. In this chapter you will learn to import a complete record layout from an external file into an unmapped data file.

Objectives

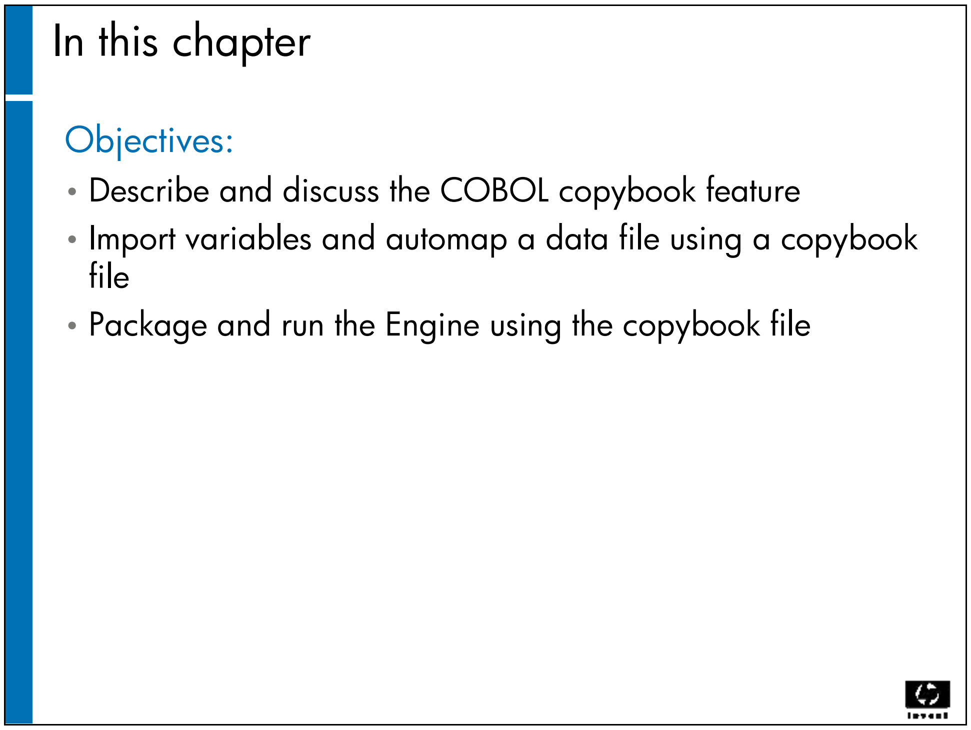
By the end of this chapter you should be able to:

- Describe and discuss the COBOL Copybook feature.
- Import variables and automap a data file using a copybook file.
- Package and run the Engine using the copybook file.

For more information

For more information, refer to the following Dialogue documentation.

▪ Data Files guide	DataFile.pdf
▪ System Administration guide	System.pdf

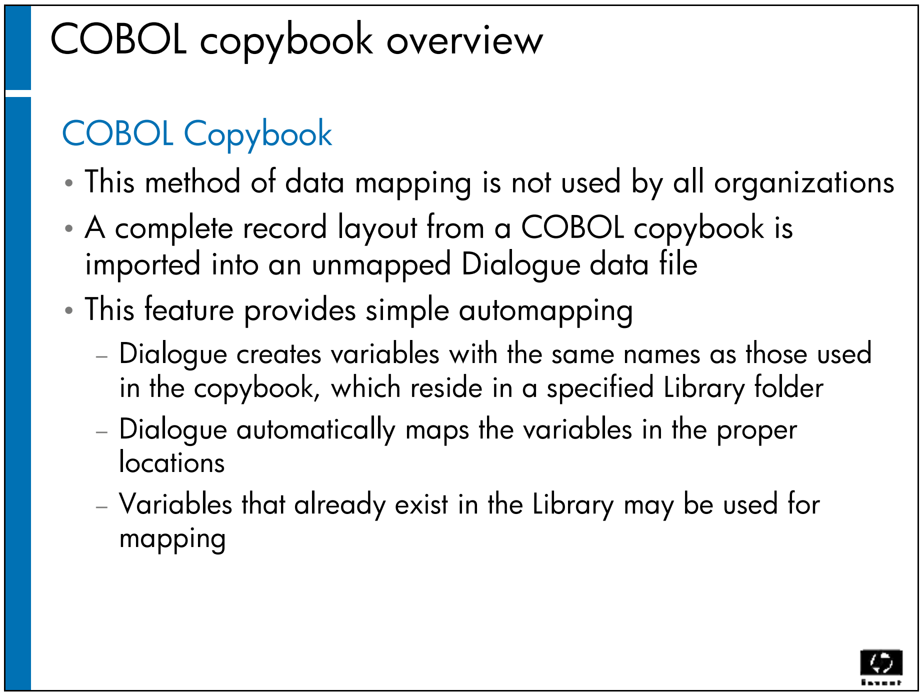


In this chapter

Objectives:

- Describe and discuss the COBOL copybook feature
- Import variables and automap a data file using a copybook file
- Package and run the Engine using the copybook file

6.2 COBOL Copybook Overview

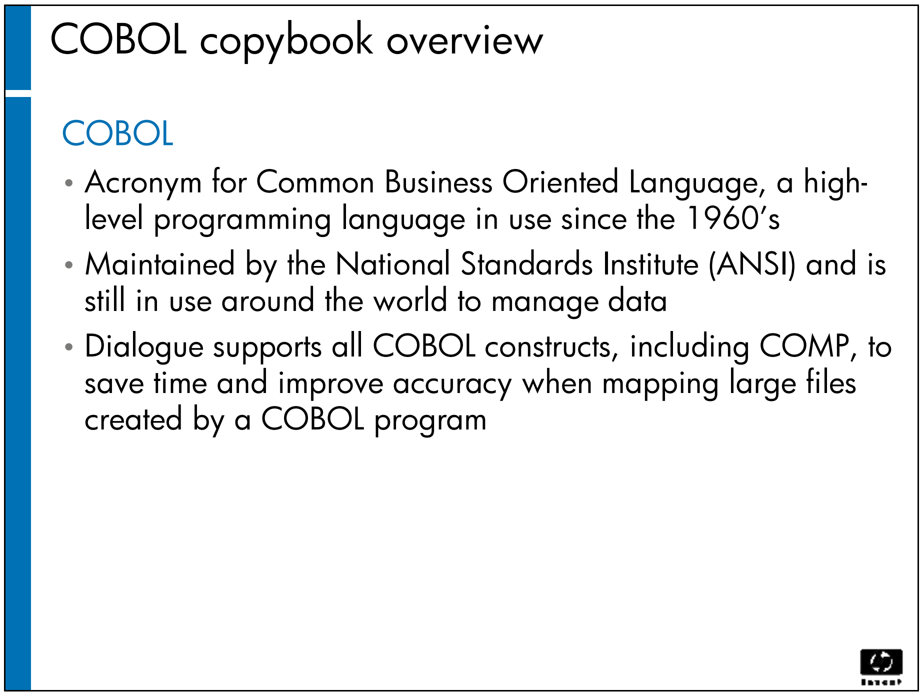


COBOL copybook overview

COBOL Copybook

- This method of data mapping is not used by all organizations
- A complete record layout from a COBOL copybook is imported into an unmapped Dialogue data file
- This feature provides simple automapping
  - Dialogue creates variables with the same names as those used in the copybook, which reside in a specified Library folder
  - Dialogue automatically maps the variables in the proper locations
  - Variables that already exist in the Library may be used for mapping

The benefit of this feature is a considerable reduction in development time. However, this does not mean that all variables will be mapped exactly as you need them. Some human intervention in the properties may be required to achieve the results you need.



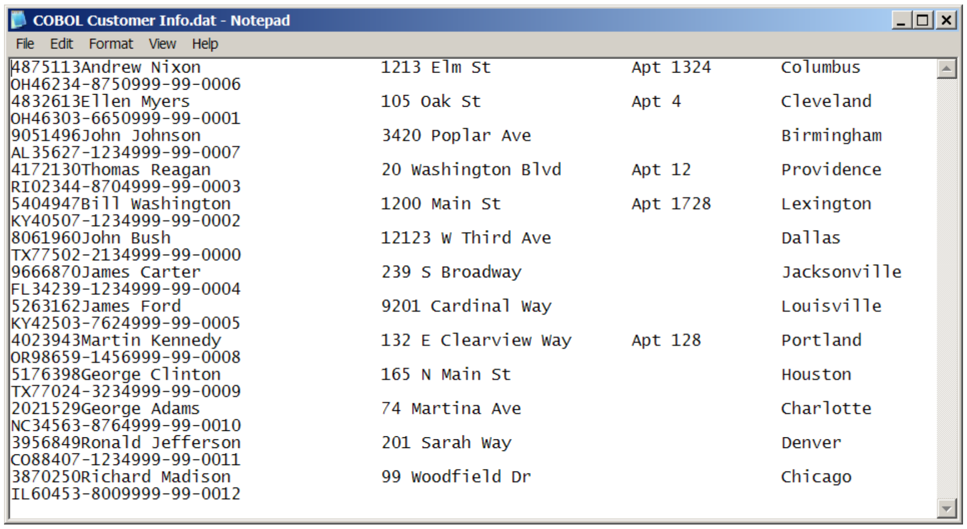
COBOL copybook overview

COBOL

- Acronym for Common Business Oriented Language, a high-level programming language in use since the 1960's
- Maintained by the National Standards Institute (ANSI) and is still in use around the world to manage data
- Dialogue supports all COBOL constructs, including COMP, to save time and improve accuracy when mapping large files created by a COBOL program

6.3 Mapping COBOL Copybook

COBOL Copybook File in a Text Editor



COBOL Customer Info.dat - Notepad

```
4875113Andrew Nixon                    1213 Elm St              Apt 1324       Columbus
OH46234-8750999-99-0006
4832613Ellen Myers                     105 Oak St               Apt 4          Cleveland
OH46303-6650999-99-0001
9051496John Johnson                    3420 Poplar Ave                         Birmingham
AL35627-1234999-99-0007
4172130Thomas Reagan                   20 Washington Blvd       Apt 12         Providence
RI02344-8704999-99-0003
5404947Bill Washington                 1200 Main St             Apt 1728       Lexington
KY40507-1234999-99-0002
8061960John Bush                       12123 W Third Ave                       Dallas
TX77502-2134999-99-0000
9666870James Carter                    239 S Broadway                          Jacksonville
FL34239-1234999-99-0004
5263162James Ford                      9201 Cardinal Way                       Louisville
KY42503-7624999-99-0005
4023943Martin Kennedy                  132 E Clearview Way      Apt 128        Portland
OR98659-1456999-99-0008
5176398George Clinton                  165 N Main St                           Houston
TX77024-3234999-99-0009
2021529George Adams                    74 Martina Ave                          Charlotte
NC34563-8764999-99-0010
3956849Ronald Jefferson                201 Sarah Way                           Denver
CO88407-1234999-99-0011
3870250Richard Madison                 99 Woodfield Dr                         Chicago
IL60453-8009999-99-0012
```

You must begin with a data file that:

- Is columnar (*not* delimited).

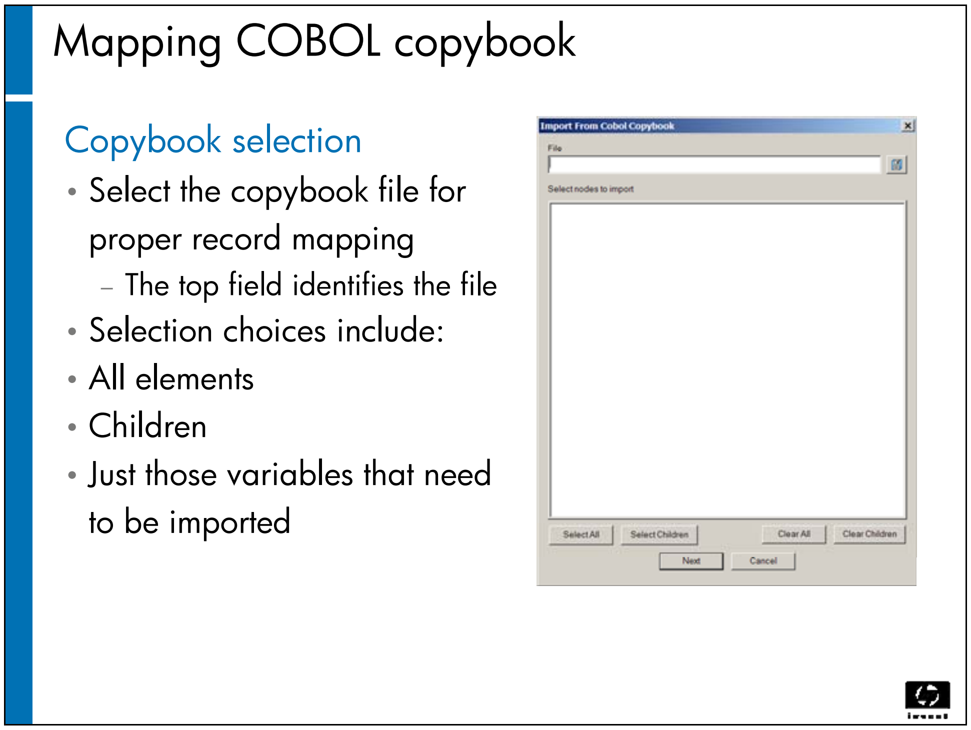
To start the copybook process you:

- Drag an unmapped data file to the Edit Panel.
- Highlight a portion of the record to map.
- Right-click and choose **Record** from the shortcut menu.
- Choose **Import from Copybook** from the submenu.

The Import from COBOL Copybook dialog box appears.

Choose the copybook file and then choose which elements to map.

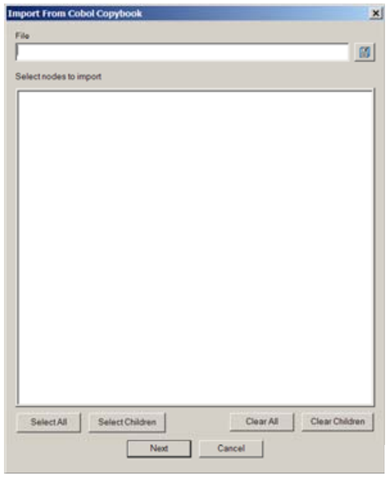
Import and Automap



Mapping COBOL copybook

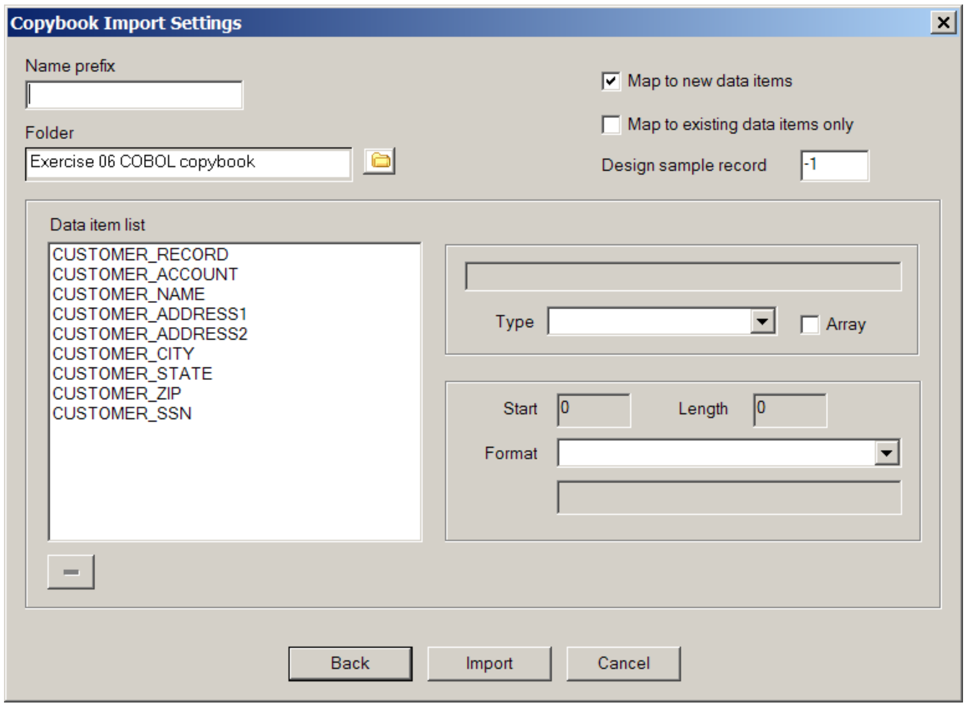
Copybook selection

- Select the copybook file for proper record mapping
  - The top field identifies the file
- Selection choices include:
- All elements
- Children
- Just those variables that need to be imported



After you press **Next**, the next dialog box to appear is the **Copybook Import Settings** dialog box.

Copybook Import Settings Dialog Box



Name prefix:

Folder: Exercise 06 COBOL copybook

[✓] Map to new data items

[ ] Map to existing data items only

Design sample record: -1

Data item list:

CUSTOMER_RECORD
CUSTOMER_ACCOUNT
CUSTOMER_NAME
CUSTOMER_ADDRESS1
CUSTOMER_ADDRESS2
CUSTOMER_CITY
CUSTOMER_STATE
CUSTOMER_ZIP
CUSTOMER_SSN

Type: [ ] Array

Start: 0 Length: 0

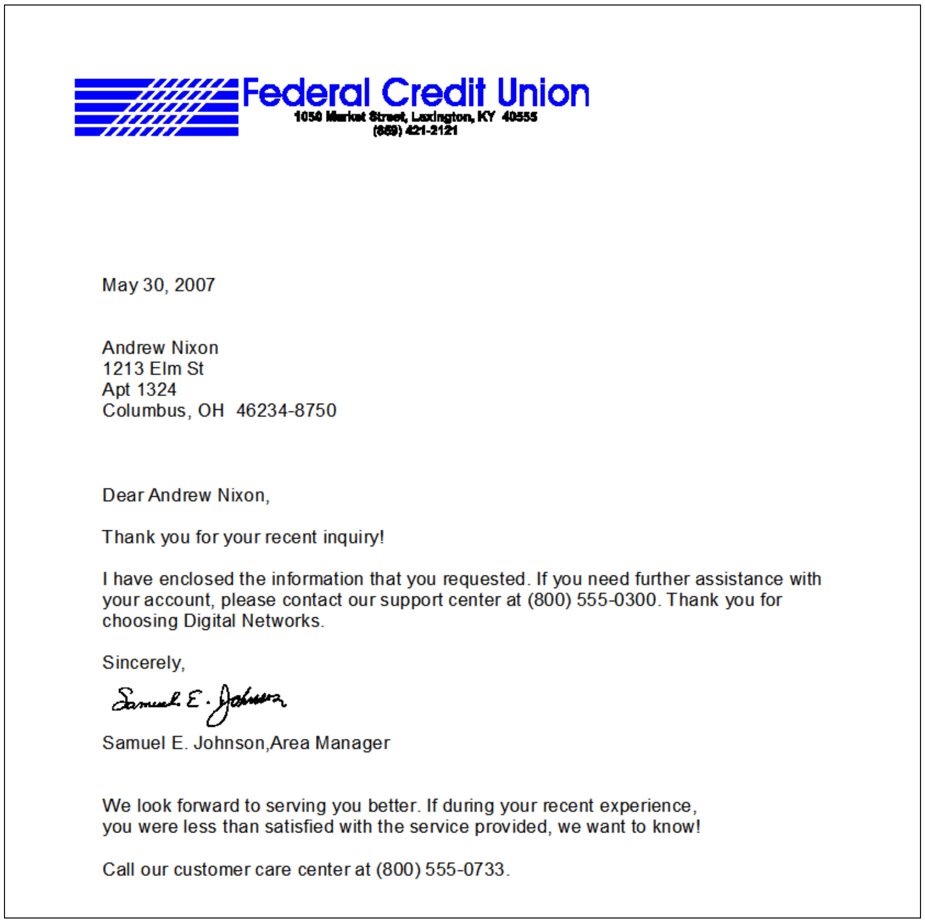
Format:

- By default, Dialogue creates new variables with the names of the variables in the copybook. If you want, you can specify a **Name Prefix** that will appear in the front of each variable name, to differentiate these variables from others in the Library.
- In the **Folder** text box you specify the folder where the new variables should reside in the Library.
- At the top, on the right side of the dialog box, select **if** and **how** you want to map these variables:
  - You can **clear** both of these check boxes.
    This will only **create** the variables (if they don't already exist) but generate no data mappings.
  - If you want to automatically map the same sequence of data items, **select** one of the following: (You cannot select both of these options.)
    - **Map To New Data Items –** this will create new variables.
    - **Map To Existing Data Items Only –** this will use variables created previously, either manually or in a previous **Copybook** execution.

6.4 Exercise: Use Dialogue COBOL Copybook

In this exercise you will use the Dialogue COBOL Copybook feature to automatically create variables and map a Customer driver file. You will put the newly created variables into a page in Designer. Then the Engine will use this Customer driver file to compose a letter for thirteen customers.

Composed Page



Federal Credit Union
1050 Market Street, Lexington, KY 40555
(859) 421-2121

May 30, 2007

Andrew Nixon
1213 Elm St
Apt 1324
Columbus, OH 46234-8750

Dear Andrew Nixon,

Thank you for your recent inquiry!

I have enclosed the information that you requested. If you need further assistance with your account, please contact our support center at (800) 555-0300. Thank you for choosing Digital Networks.

Sincerely,

Samuel E. Johnson

Samuel E. Johnson,Area Manager

We look forward to serving you better. If during your recent experience, you were less than satisfied with the service provided, we want to know!

Call our customer care center at (800) 555-0733.

A. Create a Customer Driver File

1. Expand the **Exercise 06 COBOL Copybook** folder and right-click the **Data Files** heading.
2. Choose **New Data File** from the shortcut menu.
3. The **New Data File** dialog box appears.
4. Type `Copybook` for the **Name**.
5. Choose **Customer driver file** for the **File type**.
6. Choose **Columnar data file** for the **File format**.
7. Click **Finish**.

Your new file appears in the Property Panel for you to define.

B. Define the Customer Driver File Properties

Each customer has one record in this file.

1. Click the Test Data Source tab.
2. In the **File to use for data mapping** property, type (or browse to): `C:\210 Advanced Data Concepts\Data Files\COBOL Customer Info.dat`
3. Click the **Production Data Source** tab.
4. In the **File to use in Production** field, enter: `EXdriver`
5. **Save** (but do not exit) the Property Panel.

C. Import From Copybook

1. Drag the **Copybook** data file from the Property Panel to the Edit Panel.
2. Highlight a portion of the first record by dragging over it. Dialogue gives the field a blue color.
3. Right-click to access the shortcut menu.
4. Choose **Record, Import from Copybook**.
   The Import from Cobol Copybook dialog box appears.
5. Click the button in the upper right. In the **Open** dialog box, find: `C:\210 Advanced Data Concepts\Data Files\COBOL Copybook.txt`
6. Click **Open**.
   The file appears in the dialog box, with the COBOL record and field hierarchy displayed.
7. Select the last eight headings in the window, one at a time. Although you *can* select all the record mappings, you should select only the ones you need.

8. Click **Next**.
   The **Copybook Import Settings** dialog box appears.

D. Copybook Import Settings

NOTE: Optionally, you can type a **Name Prefix** to add to all the new variables. This prefix can help you associate the variables at a glance to a particular application, vendor, client, date, or other designation.

1. Click the 🗀 button and specify the **Exercise 06 COBOL Copybook** as the folder where the new variables will reside in the Library.
2. Check the **Map to new data items** check box.

NOTE: You can highlight a variable in the **Data Item List** and specify its formatting in properties on the right, but for this exercise, this is not needed.

3. Click **Import** to import the variables and map the file.
   In the Edit Panel, note that all record data areas have a teal color to indicate they are mapped as input areas.
4. Move your cursor across the Edit Panel to read the pop-ups. Right-click and select **View** > **View Layout** to acquaint yourself with the names of the variables Dialogue created and mapped to this file.
5. Save and close the file.

E. Add New Variables to a Design Page

1. In the Library, under the **Exercise 06 COBOL Copybook** folder, expand the **Pages** heading.
2. Drag the **COBOL Page** to the Edit Panel so it will open in Designer.
   Customer variable names are typed in red text.
3. Ensure the **Variable Panel** is filtered to the **Exercise 06 COBOL Copybook** folder.
4. Replace the red variable names with the new COBOL import variables.
   - CUSTOMER_NAME
   - CUSTOMER_ADDRESS1
   - CUSTOMER_ADDRESS2
   - CUSTOMER_CITY
   - CUSTOMER_STATE
   - CUSTOMER_ZIP
5. Change text color to black after replacing variables.
6. **Save** the page.

7. **Close** Designer and return to Design Manager.

F. Add the Data File to the Application

1. Under the **Exercise 06 COBOL Copybook** folder, expand the **Applications** heading and the **Data Files** heading.
2. Drag the **Copybook** data file to the **COBOL Application**.

NOTE: Ensure the Customer Driver File (Copybook) is placed after the Initialization Data File in the COBOL Application.

NOTE: The **COBOL Page** you worked on in Designer is already referenced to the **COBOL Document** in this application.

G. Package the Application

1. Package the COBOL Application.
2. View the results in the Exstream viewer.

Chapter 7: Print Miner

7.1 In this Chapter

The **Print Miner** module lets you use an existing print file as input to Dialogue. In this chapter you will extract variable information from an existing print file and use it to compose a letter.

Objectives

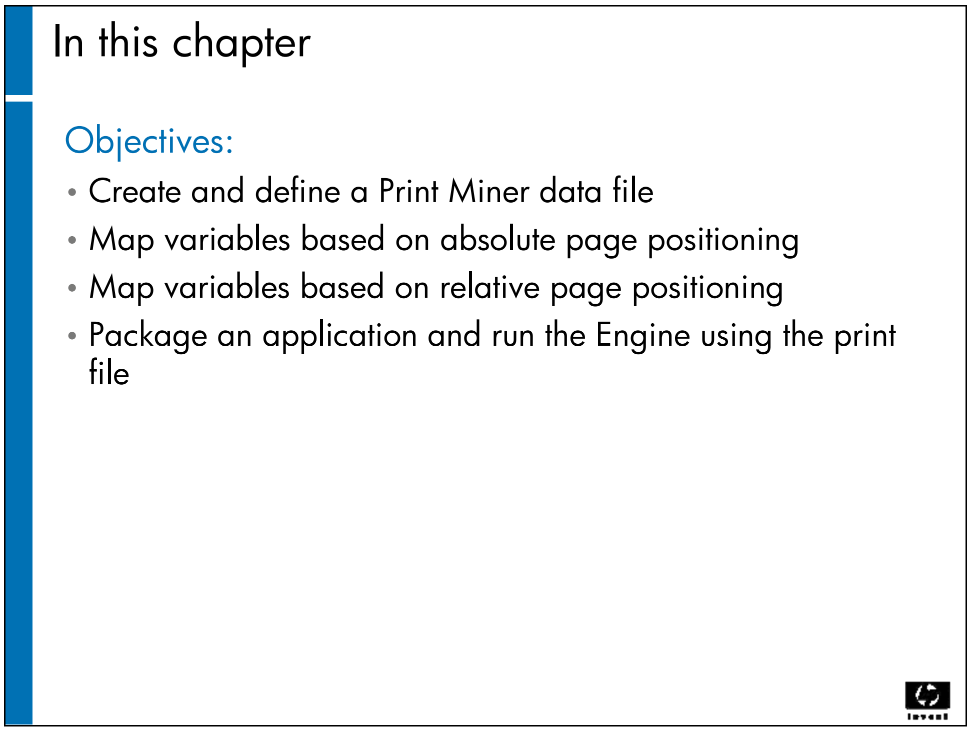
By the end of this chapter you will:

- Create and define a **Print Miner** data file.
- Map variables based on absolute page positioning.
- Map variables based on relative page positioning.
- Package and run the Engine using the print file.

For more information

For more information, refer to the following Dialogue documentation.

▪ Print Miner guide	Miner.pdf
▪ System Administration guide	System.pdf

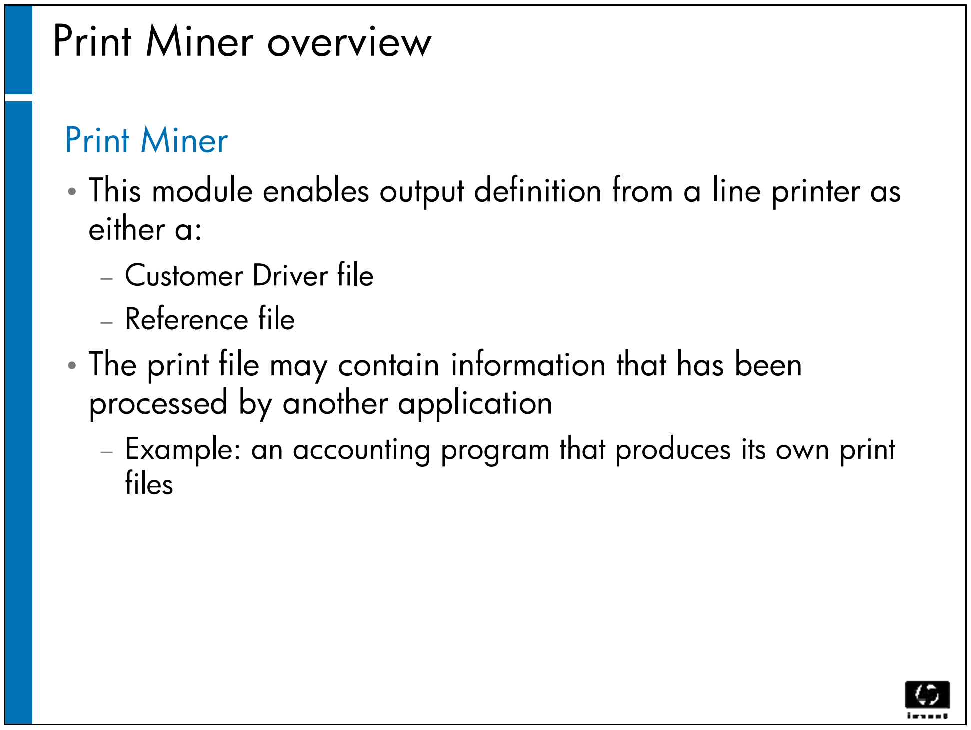


In this chapter

Objectives:

- Create and define a Print Miner data file
- Map variables based on absolute page positioning
- Map variables based on relative page positioning
- Package an application and run the Engine using the print file

7.2 Print Miner Overview



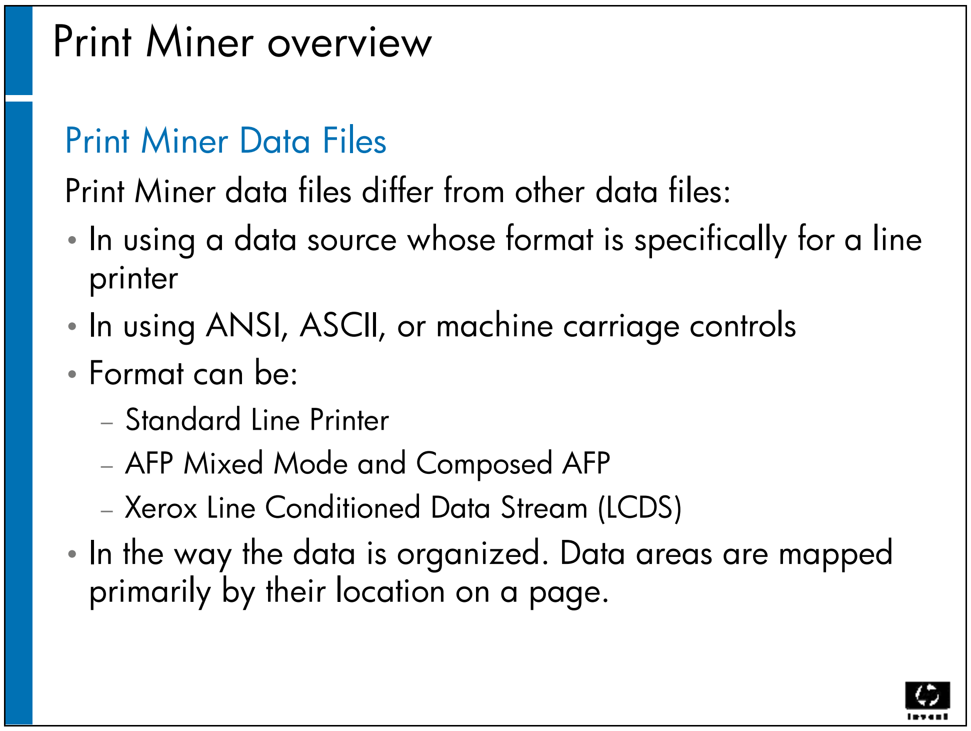
Print Miner overview

Print Miner

- This module enables output definition from a line printer as either a:
  - Customer Driver file
  - Reference file
- The print file may contain information that has been processed by another application
  - Example: an accounting program that produces its own print files

Print Miner is an add-on module for Dialogue. To verify whether you have this module, check your System Settings.

This module enables you to use one or more **print files** created by another program as a Dialogue data file.



Print Miner overview

Print Miner Data Files

Print Miner data files differ from other data files:

- In using a data source whose format is specifically for a line printer
- In using ANSI, ASCII, or machine carriage controls
- Format can be:
  - Standard Line Printer
  - AFP Mixed Mode and Composed AFP
  - Xerox Line Conditioned Data Stream (LCDS)
- In the way the data is organized. Data areas are mapped primarily by their location on a page.

Where you use *records* in other types of data files, you use *pages* in print data files.

You should understand that there are no header records in mapping a print file, but there can be header *pages.* The Engine ignores information appearing in header pages.

7.3 Line Printers

Line printers were used before the advent of laser printer technology. Early printers could only print:

- With one font
- From the top to the bottom of the page
- One line at a time

This printer produced one line from the print file and then moved the carriage based on the instructions given in the print file. These ***carriage controls*** usually appeared in the first column of the print file.

ANSI Carriage Controls

The first set of carriage controls to come into use is called **ANSI** carriage controls. They are blank, zero, dash, plus sign, numbers 1-9, and letters A-C, giving only 16 possible characters.

ANSI Carriage Controls

Hexadecimal	**Function**
X'40' (blank)	Space 1 line, then print (single line spacing)
X'F0' (zero)	Space 2 lines, then print (double line spacing)
X'60' (dash)	Space 3 lines, then print (triple line spacing)
X'4E' (plus sign)	No line spacing (Overstrike)
X'7F1' (1)	Print at line position defined as Channel 1 in the File Control Block. (FCB – an information block that will not print, but defines the codes associated with the file.) – This is the first line of a new page.
X'F2' (2)	Print at line position defined as Channel 2 in the FCB
X'F3' (3)	Print at line position defined as Channel 3 in the FCB
X'F4' (4)	Print at line position defined as Channel 4 in the FCB
X'F5' (5)	Print at line position defined as Channel 5 in the FCB
X'F6' (6)	Print at line position defined as Channel 6 in the FCB
X'F7' (7)	Print at line position defined as Channel 7 in the FCB
X'F8' (8)	Print at line position defined as Channel 8 in the FCB
X'F9' (9)	Print at line position defined as Channel 9 in the FCB
X'C1' (A)	Print at line position defined as Channel 10 in the FCB
X'C2' (B)	Print at line position defined as Channel 11 in the FCB
X'C3' (C)	Print at line position defined as Channel 12 in the FCB
Only these ANSI codes are recognized by PSF (Print Services Facility). Other codes are ignored and the text prints on the current line in single-spaced format.	

Machine Carriage Controls

IBM created a more complex set of carriage controls called **Machine** carriage controls. This set uses hexadecimal codes: two digits represent 0 to 255 and thus, there are 256 characters possible.

Later, line printers contained multiple print heads, supporting up to four fonts. This involved the addition of a font column to control the multiple heads.

Machine Carriage Controls

Hexadecimal Characters	**Function**
X’03’	No operation
X’09’	Print and space 1 line (single line spacing)
X’11’	Print and space 2 lines (double line spacing)
X’19’	Print and space 3 lines (triple line spacing)
X’01’	No line spacing (Overstrike)
X’89’	Print, then skip to line position defined as Channel 1 in the File Control Block (FCB) – first line of a new page
X’91’	Print, then skip to line defined as Channel 2 in the FCB
X’99’	Print, then skip to line defined as Channel 3 in the FCB
X’A1’	Print, then skip to line defined as Channel 4 in the FCB
X’A9’	Print, then skip to line defined as Channel 5 in the FCB
X’B1’	Print, then skip to line defined as Channel 6 in the FCB
X’B9’	Print, then skip to line defined as Channel 7 in the FCB
X’C1’	Print, then skip to line defined as Channel 8 in the FCB
X’C9’	Print, then skip to line defined as Channel 9 in the FCB
X’D1’	Print, then skip to line defined as Channel 10 in the FCB
X’D9’	Print, then skip to line defined as Channel 11 in the FCB
X’E1’	Print, then skip to line defined as Channel 12 in the FCB
X’0B’	Space 1 line without printing
X’13’	Space 2 lines without printing
X’1B’	Space 3 lines without printing
X’8B’	Skip to Channel 1 now (next page)
X’93’	Skip to Channel 2 now (as in FCB)
X’9B’	Skip to Channel 3 now (as in FCB)
X’A3’	Skip to Channel 4 now (as in FCB)
X’AB’	Skip to Channel 5 now (as in FCB)
X’B3’	Skip to Channel 6 now (as in FCB)
X’BB’	Skip to Channel 7 now (as in FCB)
X’C3’	Skip to Channel 8 now (as in FCB)
X’CB’	Skip to Channel 9 now (as in FCB)
X’D3’	Skip to Channel 10 now (as in FCB)
X’DB’	Skip to Channel 11 now (as in FCB)
X’E3’	Skip to Channel 12 now (as in FCB)

Hexadecimal Characters	**Function**
PSF (Print Services Facility) ignores the following control characters, and lines containing them do not print: X'02' – X'07' X'0A' X'12' X'23' X'43' X'63' X'6B' X'73' X'7B' X'EB' X'F3' X'FB'	
PSF treats any other codes as invalid and prints the data single-spaced on the current line.	

These two early methods are still used in line data printing. However, methods became more involved as printers became more technologically advanced.

ASCII Carriage Controls

The ASCII carriage controls are very simple. Inherently, using this method can cause occasional data errors if the printer encounters these characters within the data.

ASCII Carriage Controls

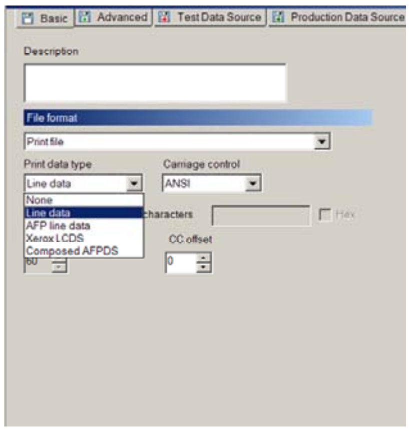
Characters	**Function**
0D	Do not space.
0D 0A	Space one line.
0C	Skip to Channel 1.

7.4 Print File Types

Print File types

Print Data Type list

- In the data file properties, new properties appear with a File format selection of **Print file**
- The following discusses the options in the Print data type list box (reading from the bottom up)



Description

File format
Print file

Print data type
Line data

None
Line data
AFP line data
Xerox LCDS
Composed AFPDS

60

Carriage control
ANSI

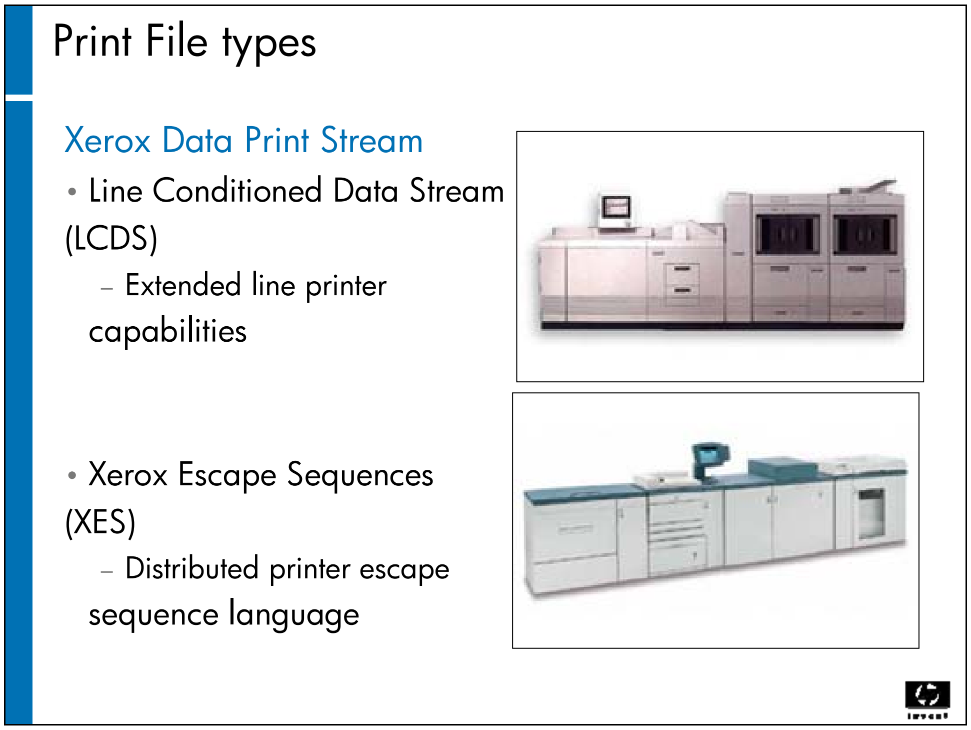
characters

Hex

CC offset
0

Xerox Line Conditioned Data Stream

The Xerox Line Conditioned Data Stream (LCDS) print file is a print file format supported in Print Miner.

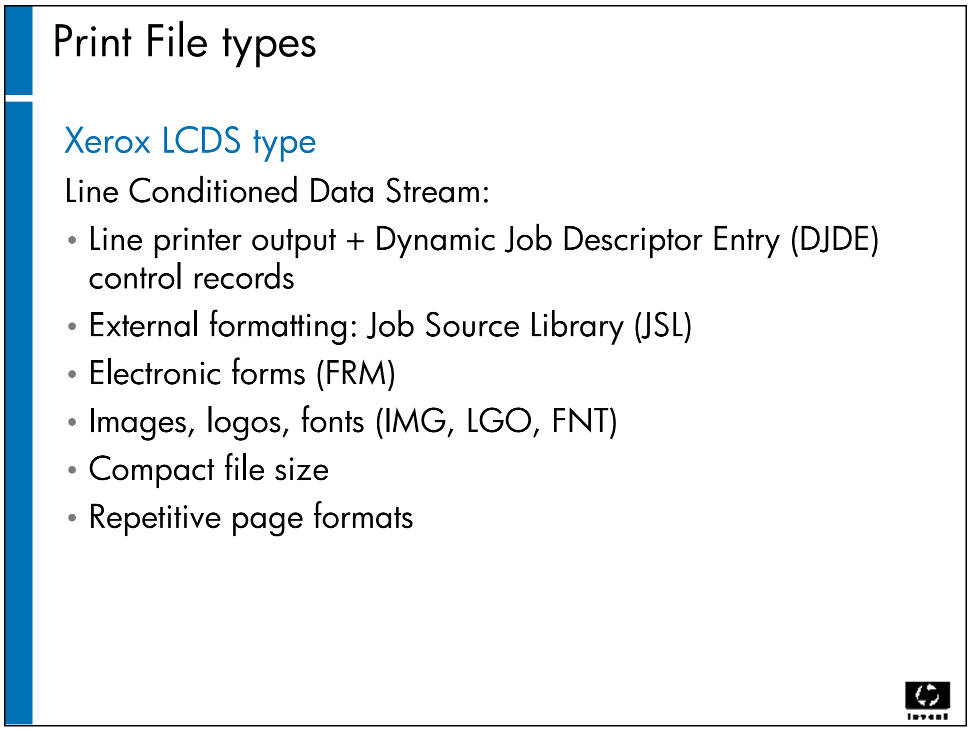


Print File types

Xerox Data Print Stream

- Line Conditioned Data Stream (LCDS)
  - Extended line printer capabilities

- Xerox Escape Sequences (XES)
  - Distributed printer escape sequence language



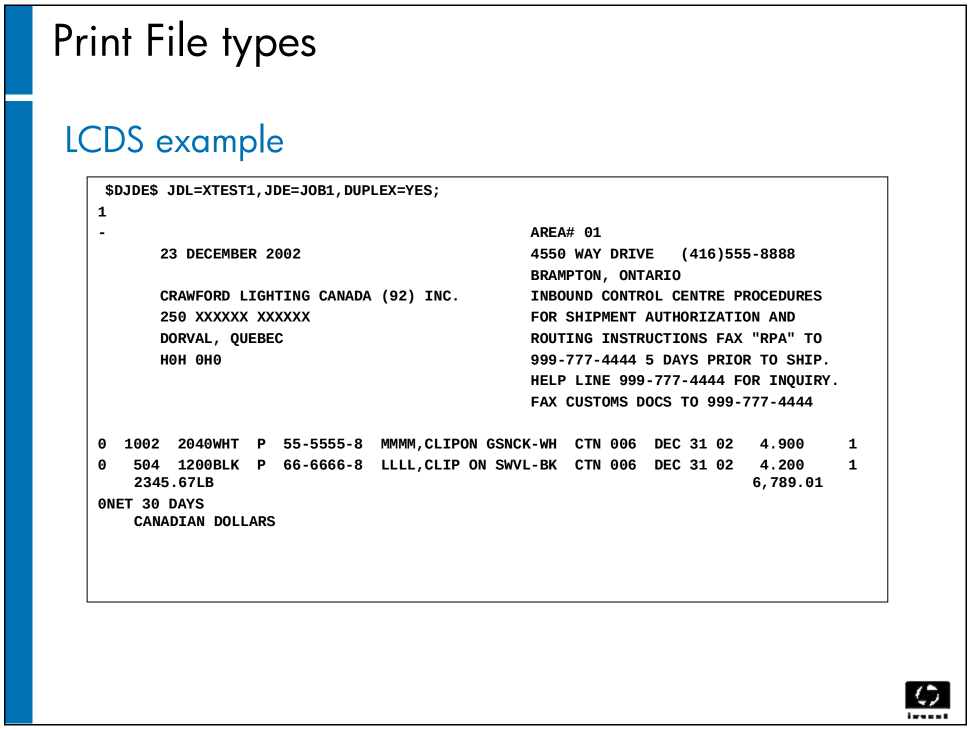
Print File types

Xerox LCDS type

Line Conditioned Data Stream:

- Line printer output + Dynamic Job Descriptor Entry (DJDE) control records
- External formatting: Job Source Library (JSL)
- Electronic forms (FRM)
- Images, logos, fonts (IMG, LGO, FNT)
- Compact file size
- Repetitive page formats

This slide shows an example of LCDS that can be mapped as data in Dialogue.



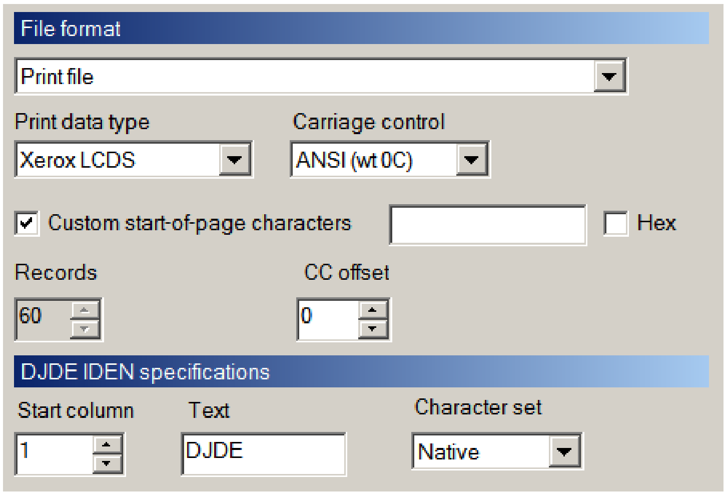
Print File types

LCDS example

```
 $DJDE$ JDL=XTEST1,JDE=JOB1,DUPLEX=YES;
1
-                                                AREA# 01
       23 DECEMBER 2002                          4550 WAY DRIVE   (416)555-8888
                                                 BRAMPTON, ONTARIO
       CRAWFORD LIGHTING CANADA (92) INC.        INBOUND CONTROL CENTRE PROCEDURES
       250 XXXXXX XXXXXX                         FOR SHIPMENT AUTHORIZATION AND
       DORVAL, QUEBEC                            ROUTING INSTRUCTIONS FAX "RPA" TO
       H0H 0H0                                   999-777-4444 5 DAYS PRIOR TO SHIP.
                                                 HELP LINE 999-777-4444 FOR INQUIRY.
                                                 FAX CUSTOMS DOCS TO 999-777-4444

0  1002  2040WHT  P  55-5555-8  MMMM,CLIPON GSNCK-WH  CTN 006  DEC 31 02   4.900     1
0   504  1200BLK  P  66-6666-8  LLLL,CLIP ON SWVL-BK  CTN 006  DEC 31 02   4.200     1
    2345.67LB                                                             6,789.01
0NET 30 DAYS
    CANADIAN DOLLARS
```

If you choose Xerox LCDS as the Print data type, new options appear under DJDE IDEN Specifications:

Data File Properties - File Format Area



File format

Print file

Print data type: Xerox LCDS

Carriage control: ANSI (wt 0C)

[✓] Custom start-of-page characters [ ] Hex

Records: 60

CC offset: 0

DJDE IDEN specifications

Start column: 1

Text: DJDE

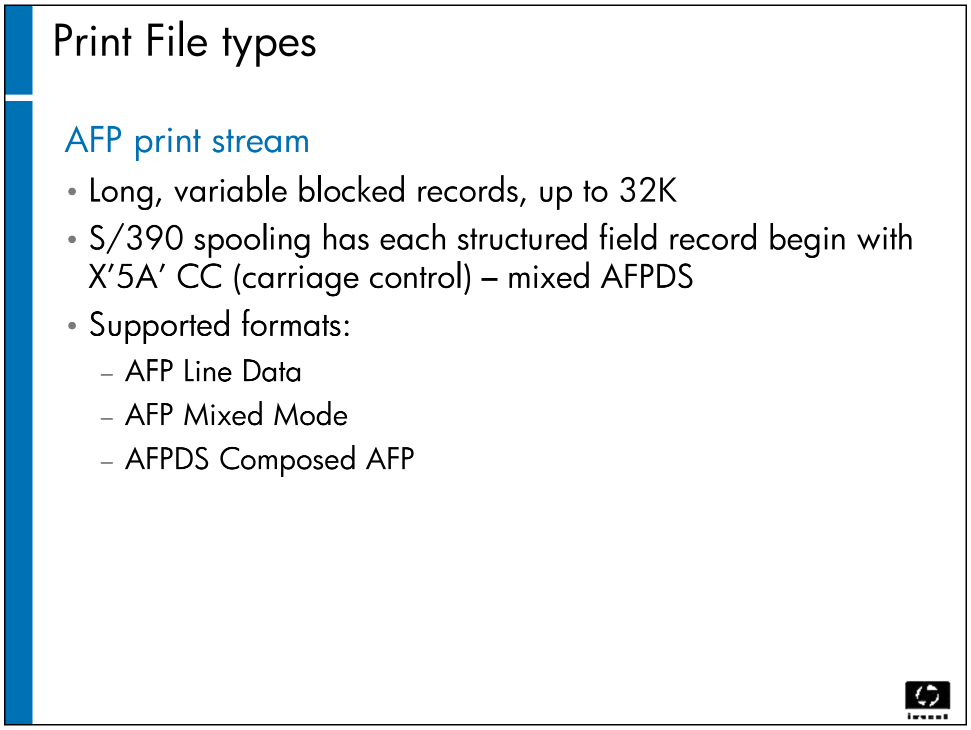
Character set: Native

- Under **Carriage control**, choose **ANSI (wt 0C)** (ANSI character plus 1), **ANSI** (16 characters), **ASCII** (seven characters), or **Machine** (256 characters).
- The **Custom start-of-page characters** check box enables you to specify custom characters that Print Miner uses to recognize the beginning of a page. You can specify up to 256 characters in ASCII, EBCDID, or hexadecimal format.

- Below **Start column**, enter the number of columns the identification string is offset from the first column, or position 0. For example, you select 1 if the data appears in the second column. This data does not appear in the first column, so 0 is not a valid selection.
- In the **CC Offset** box, specify the offset to the column that contains the carriage control. Normally this is position **0**, or the first column, but it could be elsewhere
- Type the **Text** used as the IDEN.
- Choose a character set from the drop-down list. The options are **Native**, **ASCII**, or **EBCDIC**.

AFP Line Data

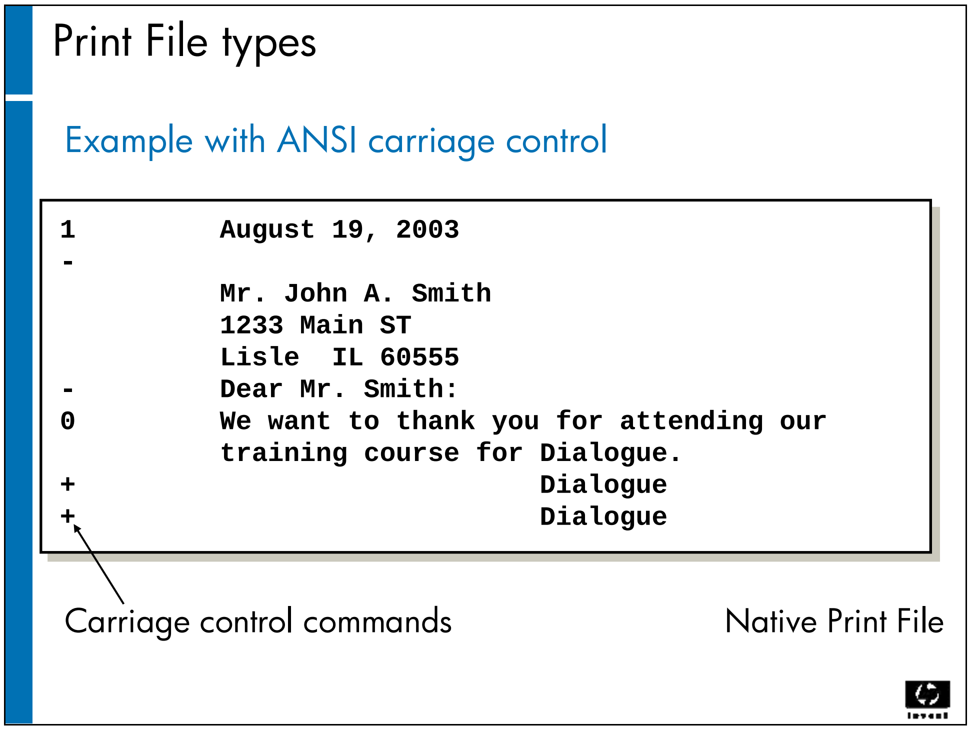
Choose **AFP Line Data** as the **Print data type** when your organization will use AFP output conforming to this standard.



Print File types

AFP print stream

- Long, variable blocked records, up to 32K
- S/390 spooling has each structured field record begin with X'5A' CC (carriage control) – mixed AFPDS
- Supported formats:
  - AFP Line Data
  - AFP Mixed Mode
  - AFPDS Composed AFP



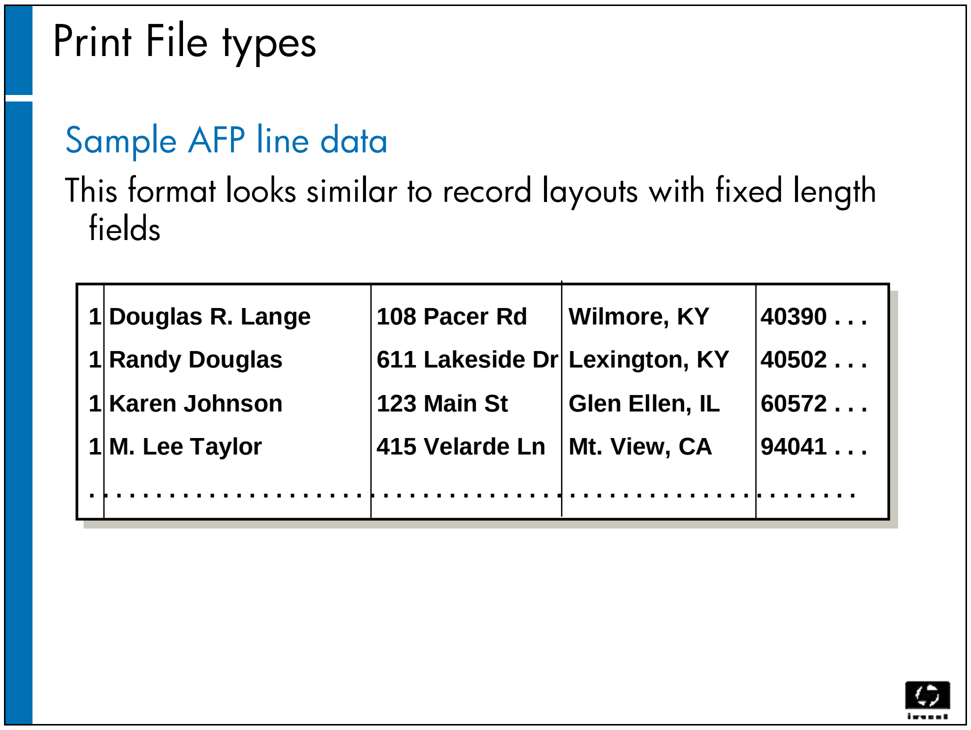
Print File types

Example with ANSI carriage control

```
1         August 19, 2003
-
          Mr. John A. Smith
          1233 Main ST
          Lisle  IL 60555
-         Dear Mr. Smith:
0         We want to thank you for attending our
          training course for Dialogue.
+                             Dialogue
+                             Dialogue
```

Carriage control commands

Native Print File

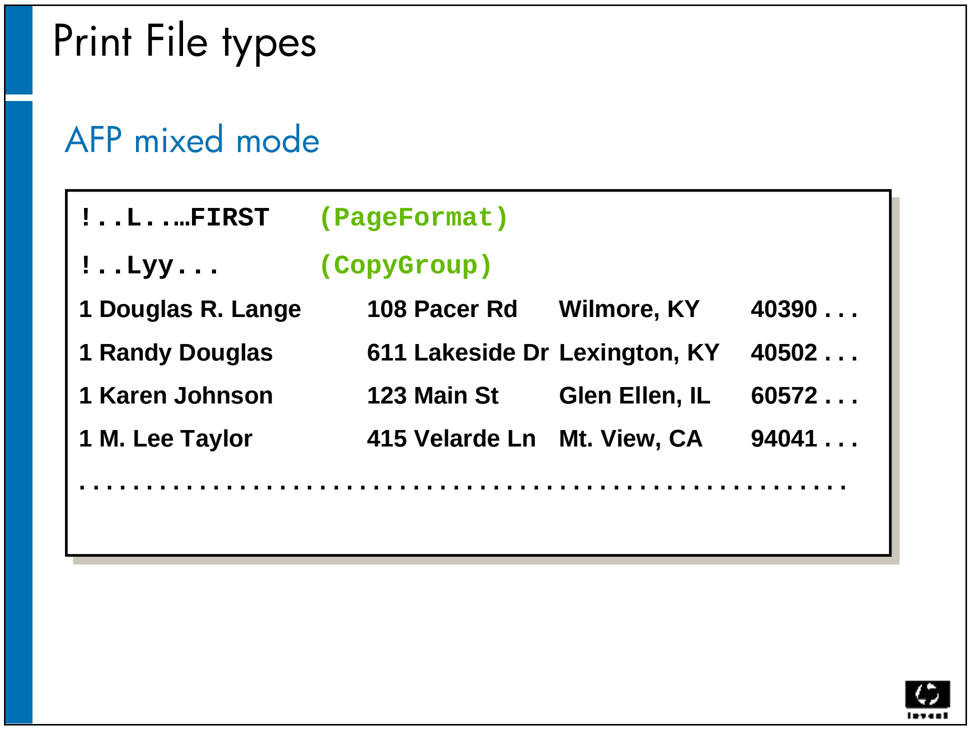


Print File types

Sample AFP line data

This format looks similar to record layouts with fixed length fields

1	**Douglas R. Lange**	**108 Pacer Rd**	**Wilmore, KY**	**40390 . . .**
1	**Randy Douglas**	**611 Lakeside Dr**	**Lexington, KY**	**40502 . . .**
1	**Karen Johnson**	**123 Main St**	**Glen Ellen, IL**	**60572 . . .**
1	**M. Lee Taylor**	**415 Velarde Ln**	**Mt. View, CA**	**94041 . . .**
.	. . . .	. . . .	. . . .	. . . .



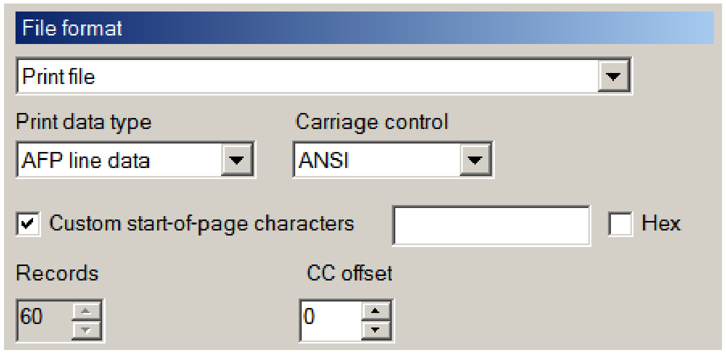
Print File types

AFP mixed mode

```
!..L.…FIRST    (PageFormat)
!..Lyy...      (CopyGroup)
1 Douglas R. Lange     108 Pacer Rd      Wilmore, KY      40390 . . .
1 Randy Douglas        611 Lakeside Dr   Lexington, KY    40502 . . .
1 Karen Johnson        123 Main St       Glen Ellen, IL   60572 . . .
1 M. Lee Taylor        415 Velarde Ln    Mt. View, CA     94041 . . .
. . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . . .
```

Define Properties

Data File Properties – File Format Area



If you choose **AFP Line Data** as the **Print data type**:

- Under **Carriage control**, choose **ANSI (wt 0c)** (ANSI character plus 1), **ANSI** (16 characters), **ASCII** (seven characters), or **Machine** (256 characters).
- The **Custom start-of-page characters** check box enables you to specify custom characters that Print Miner uses to recognize the beginning of a page. You can specify up to 255 characters in ASCII, EBCDID, or hexadecimal format.
- Enter the **CC Offset**, the offset to the column that contains the carriage control. Normally this is position **0**, or the first column, but it could be elsewhere.

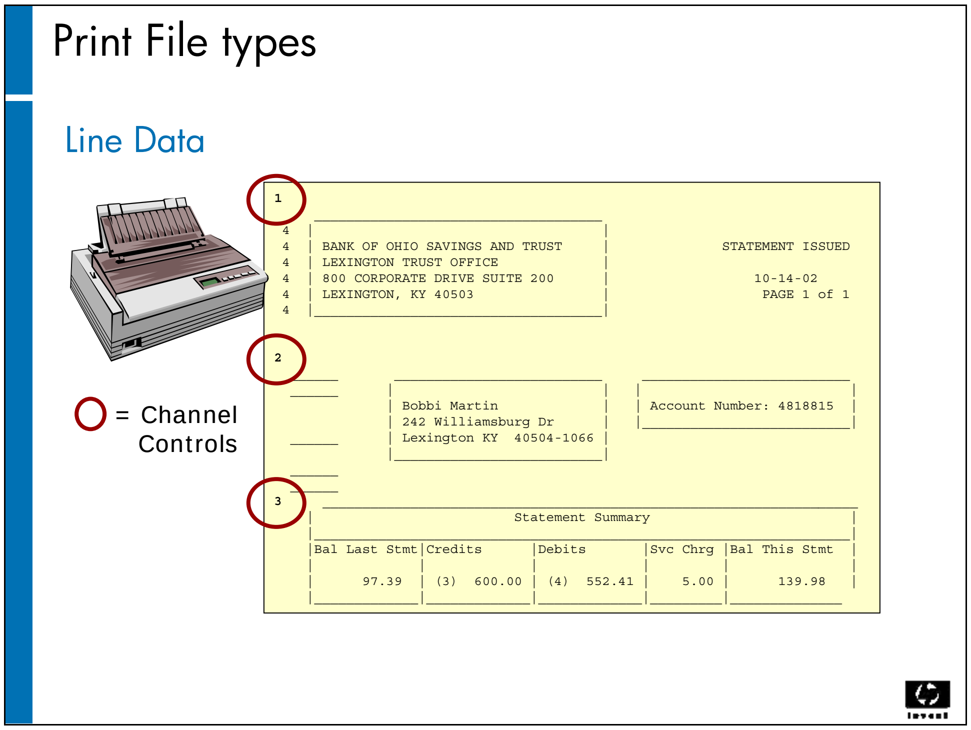
Composed AFPDS

With this **Print data type** option Dialogue converts a fully composed AFP file into line data, which you can then use with Print Miner.

Note: Refer to the *Print Miner* guide for more information on options specific to Composed AFPDS.

Line Data

This **Print data type** option is used for generic **Line Data** that adheres to general conventions.



Print File types

Line Data

1
2
3
= Channel Controls

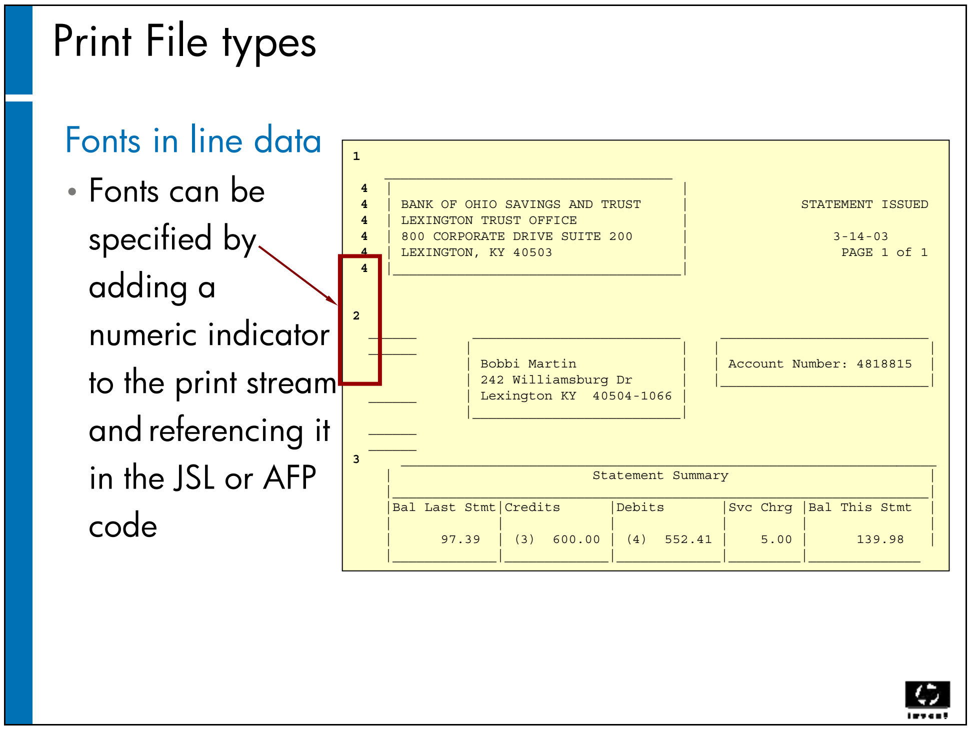
```
 1
     ____________________________________
 4  |                                    |
 4  | BANK OF OHIO SAVINGS AND TRUST     |            STATEMENT ISSUED
 4  | LEXINGTON TRUST OFFICE             |
 4  | 800 CORPORATE DRIVE SUITE 200      |               10-14-02
 4  | LEXINGTON, KY 40503                |                PAGE 1 of 1
 4  |____________________________________|

 2
 ______     ___________________________     ___________________________
 ______    |                           |   |                           |
           | Bobbi Martin              |   | Account Number: 4818815   |
           | 242 Williamsburg Dr       |   |___________________________|
 ______    | Lexington KY  40504-1066  |
           |___________________________|
 ______
 ______
 3
    _____________________________________________________________________
   |                          Statement Summary                         |
   |____________________________________________________________________|
   |Bal Last Stmt|Credits      |Debits       |Svc Chrg |Bal This Stmt   |
   |             |             |             |         |                |
   |       97.39 | (3)  600.00 | (4)  552.41 |    5.00 |       139.98   |
   |_____________|_____________|_____________|_________|______________  
```

Channel Controls

As printers became more sophisticated, it becomes possible to "pre-program" specific line positions for the print head to find when it saw a specific carriage control command. This allows the programmer to pre-define positions on pre-printed forms and tell the print head to move to them without trying to count how many lines the printer had printed, and thus how many it had to move.

These controls are sometimes called channel controls and generally have the values of 2, 3, 4, … C. IBM uses a file called a Forms Control Buffer (FCB) to tell the printer the position to associate with each of these values.

Fonts in Line Data



Print File types

Fonts in line data

- Fonts can be specified by adding a numeric indicator to the print stream and referencing it in the JSL or AFP code

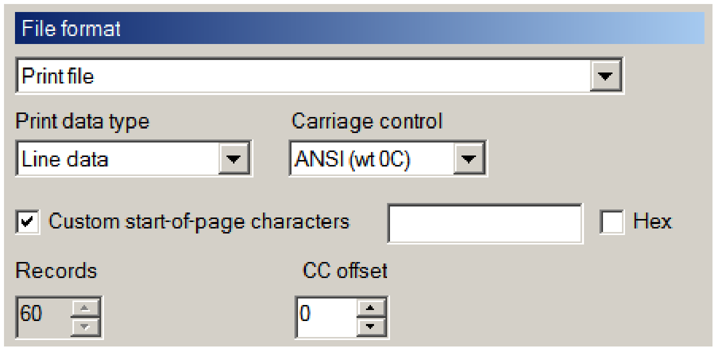
```
1

4  |                                   |
4  | BANK OF OHIO SAVINGS AND TRUST    |          STATEMENT ISSUED
4  | LEXINGTON TRUST OFFICE            |
4  | 800 CORPORATE DRIVE SUITE 200     |             3-14-03
4  | LEXINGTON, KY 40503               |              PAGE 1 of 1
4  |___________________________________|

2

            | Bobbi Martin            |    | Account Number: 4818815  |
            | 242 Williamsburg Dr     |    |__________________________|
            | Lexington KY  40504-1066|
            |_________________________|

3
   |                         Statement Summary                              |
   |                                                                        |
   |Bal Last Stmt|Credits      |Debits       |Svc Chrg |Bal This Stmt       |
   |             |             |             |         |                    |
   |      97.39  | (3)  600.00 | (4)  552.41 |   5.00  |      139.98        |
   |_____________|_____________|_____________|_________|____________________|
```

Eventually, line data printers were built that could handle multiple fonts. A specific font could be chosen for each line of print by specifying a font number for the line. Normally, this font information is placed in column two of the print stream (with column one having carriage control information). The actual font associated with the font number was specified through the JCL or other means.

In older printers, there were limits on the total number of fonts that could be used, generally four.

Define Properties

Data File Properties - File Format Area



File format

Print file

Print data type: Line data

Carriage control: ANSI (wt 0C)

☑ Custom start-of-page characters ☐ Hex

Records: 60

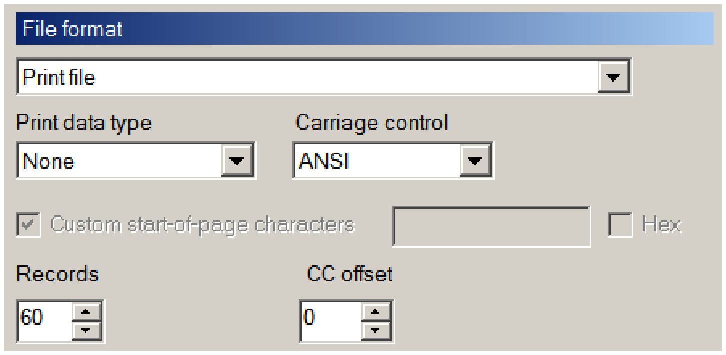
CC offset: 0

- Under **Carriage control**, choose **ANSI (wt 0c)** (ANSI character plus 1), **ANSI** (16 characters), **ASCII** (seven characters), or **Machine** (256 characters).
- The **Custom start-of-page characters** check box enables you to specify custom characters that Print Miner uses to recognize the beginning of a page. You can specify up to 255 characters in ASCII, EBCDID, or hexadecimal format.
- Enter the **CC Offset**, the offset to the column that contains the carriage control. Normally this is position **0**, or the first column, but it could be elsewhere.

None

When you choose the **None** option, the file is a plain text file and has no special formatting. You must type the number of lines (records) per page.

Data File Properties – File Format Area



File format

Print file

Print data type: None

Carriage control: ANSI

Custom start-of-page characters

Hex

Records: 60

CC offset: 0

If you choose **None** in the **Print data type** drop-down list:

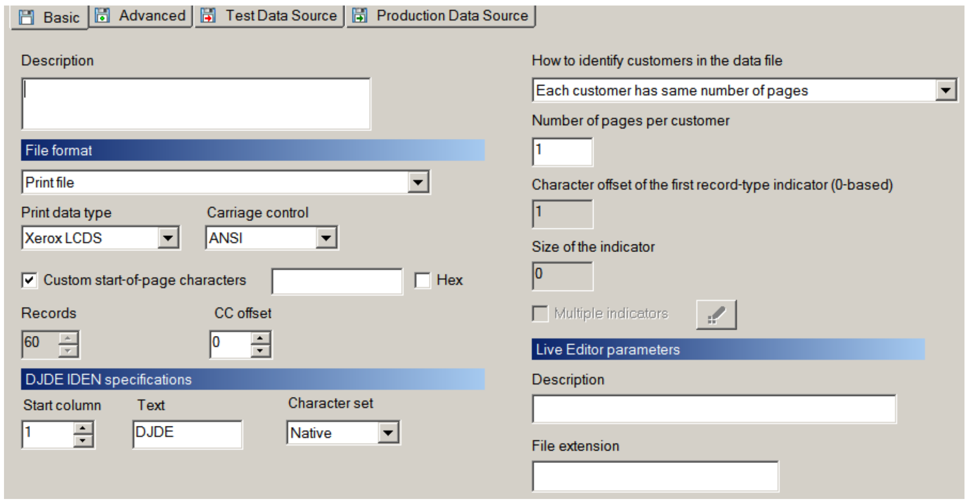
- Under **Carriage control**, choose **ANSI (wt 0c)** (ANSI character plus 1), **ANSI** (16 characters), **ASCII** (seven characters), or **Machine** (256 characters).
- Type the number of lines (records) per page in the **Records** text box.
- Enter the **CC Offset**, the offset to the column that contains the carriage control. Normally this is position **0**, or the first column, but it could be elsewhere.

7.5 Print File Properties

Basic Tab

The following discusses the How to identify customers in the data file area on the Basic tab.

Data File Properties – Basic Tab



Description:
Print file
Print data type: Xerox LCDS
Carriage control: ANSI
[✓] Custom start-of-page characters
[ ] Hex
Records: 60
CC offset: 0
Start column: 1
Text: DJDE
Character set: Native
How to identify customers in the data file: Each customer has same number of pages
Number of pages per customer: 1
Character offset of the first record-type indicator (0-based): 1
Size of the indicator: 0
[ ] Multiple indicators
Description:
File extension:

From the How to identify customers in the data file drop-down list, specify how pages are mapped in the data file. Options are:

- **Each customer has same number of pages** - Each customer in the data file has the same number of pages. This is the default option. When you select this option, enter the number of pages for each customer in the **Number of pages per customer** box.
- **A new customer occurs on specified page type(s)** - Customers do not always have the same number of pages. When you select this option, specify the:
  - Beginning location of the first page type indicator in the **Location of the (first) record-type indicator** box.
  - Length of the page type indicator in the Size of the indicator (use 0 if delimited and want entire field) box.

NOTE: The choice you make here is important to the way that your pages will be mapped in the data file. Review the information in your data file. If **each customer has the same number of pages**, you should choose that option.

Advanced Tab

You set the general data file properties on the **Advanced** tab the same way as you do with other data file formats. The exception is that you can set the number of **Header Pages** found in the file.

If you have header pages in the print file:

- Type the number of **Header** pages in the file.
- Select the **Pages** check box.

This will speed processing over defining the headers by record.

Data Source Tabs

Define the **File Names** as you would for other data files.

- Enter the **File to use for data mapping**, the name of the print file you are using for the data input file, or browse for the file.
- Use the button to go to a text editor to edit the print file. You may want to do this if you have header information you want to add to the file manually.
- In the **Production Data Source** tab, enter file to use for production.

7.6 Mapping a Print File

When you drag a Print Miner data file to the Edit Panel, you will generally see the carriage controls in the first column of data. You map this file in much the same way you map other data files:

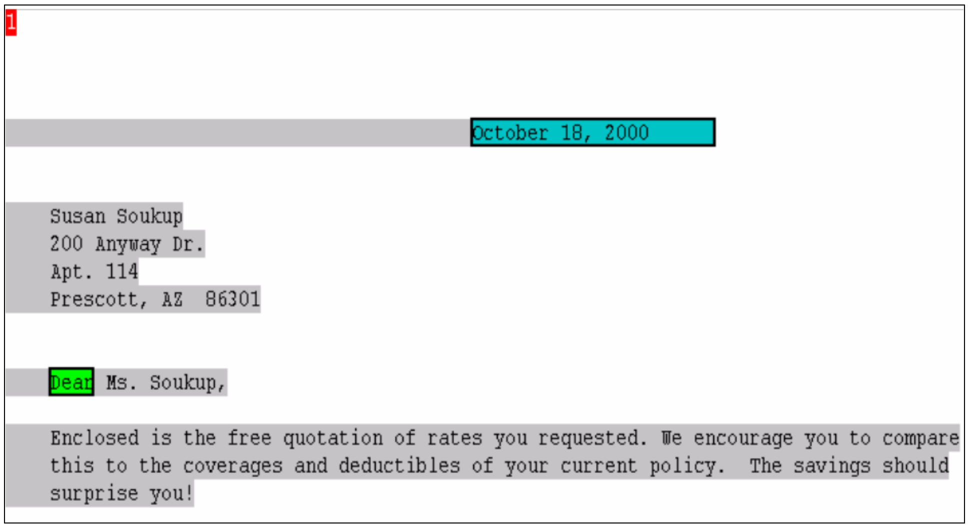
- You highlight an area.
- You specify how the data is to be recognized.

However, the data areas are described differently, largely by the location on the page where the data appears.

Print Miner files also support a unique property, called a **spot**. A **spot** is an anchor point on a page. Data can be mapped based on relative position to the spot. A spot can float, enabling mapped data to float with it. You can define any of the following to be a spot:

- Exact (static) text.
- Exact position, including a range of lines and columns (channels).
- A position relative to any page.
- A position relative to any information that appears consistently in the file (this includes other spots).

Partially Mapped Print Miner File in the Edit Panel

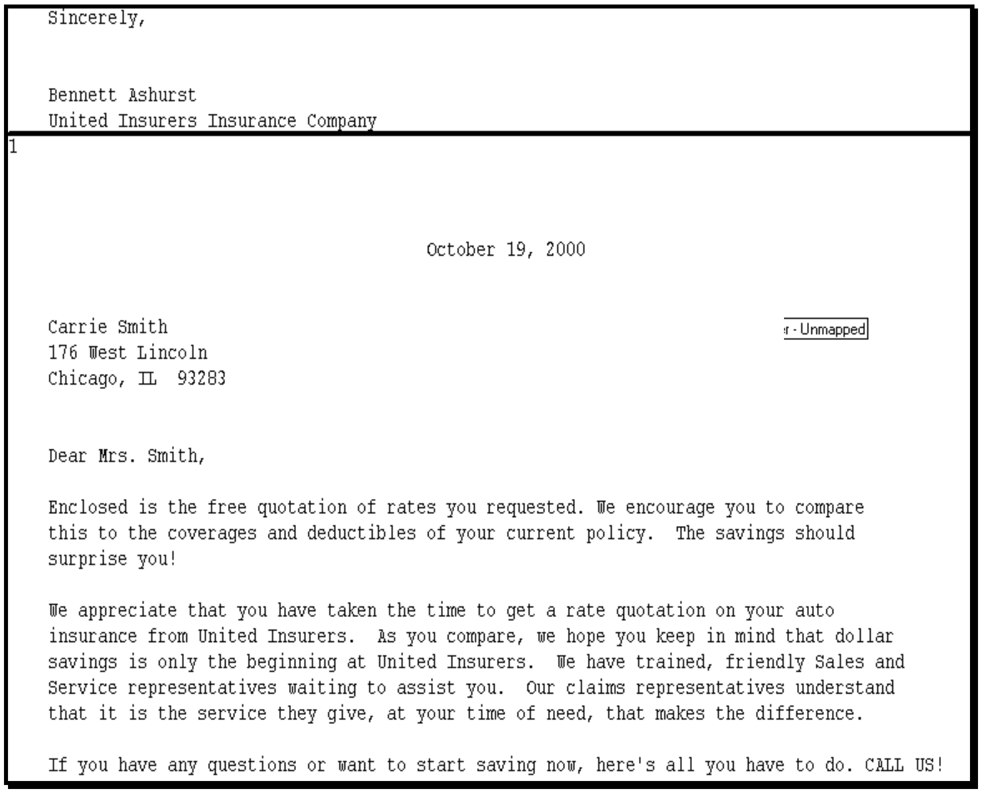


```
1

                                        October 18, 2000


    Susan Soukup
    200 Anyway Dr.
    Apt. 114
    Prescott, AZ  86301


    Dear Ms. Soukup,

    Enclosed is the free quotation of rates you requested. We encourage you to compare
    this to the coverages and deductibles of your current policy.  The savings should
    surprise you!
```

In the example above, the gray (unmapped) fields identify this page as a *header page*. The word Dear is a *spot*, highlighted in green. The date is a *data area*, highlighted in teal. The red number “1” in the upper left, top, is a *page indicator*, highlighted in red. You will learn more about these in the remainder of this chapter.

Example: Letter Page

Letter Page



```
    Sincerely,


    Bennett Ashurst
    United Insurers Insurance Company
1



                                       October 19, 2000


    Carrie Smith                                                       r - Unmapped
    176 West Lincoln
    Chicago, IL  93283


    Dear Mrs. Smith,

    Enclosed is the free quotation of rates you requested. We encourage you to compare
    this to the coverages and deductibles of your current policy.  The savings should
    surprise you!

    We appreciate that you have taken the time to get a rate quotation on your auto
    insurance from United Insurers.  As you compare, we hope you keep in mind that dollar
    savings is only the beginning at United Insurers.  We have trained, friendly Sales and
    Service representatives waiting to assist you.  Our claims representatives understand
    that it is the service they give, at your time of need, that makes the difference.

    If you have any questions or want to start saving now, here's all you have to do. CALL US!
```

This illustration shows a letter print file in the Edit Panel. The end of one page where a new customer starts is shown by the bold black line. In this case, **Each customer has the same number of pages**, and that number is **1**.

Print File Layout Compared to Printed Output

The lines that you see in the print layout file may not correspond to a printed line. The carriage controls define the printed format. Each record (line) in the layout may print on a different number of lines than you see in the file. The context-sensitive pop-up will describe the area of the file your pointer covers and the status bar will give its actual location.

Color in the Data File in the Edit Panel

Line colors help reduce the time required to map a print file.

- A **bold line** indicates the start of a new customer.
- A **dashed line** indicates a new page.
- A **red line** indicates a new section.

Page Background Colors

- **Yellow –** the page is undefined.
- **Pale Yellow –** the page satisfies more than one definition type.
- **Gray –** indicates a header page. Header pages are automatically ignored and remain gray.
- **White –** indicates a with a single page type definition.
- **Black –** indicates a page to be ignored. Header pages marked **Ignore** will be black.

Data Area Highlight Colors

Highlight colors indicate the data area type as in mapping other files in Dialogue.

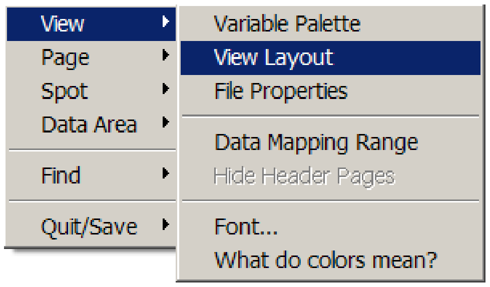
- **Teal –** indicates a data area that is used to read data (input area).
- **Aqua –** indicates an active area.
- **Green –** indicates an inactive spot. A **spot** is an absolute location on a page. Other spots can be made relative to a spot.
- **Bright Green –** indicates an **active** spot.
- **Red –** indicates a page identifier record. If you see solid red across the entire record, it indicates the end of all data areas **in sections** for this page.
- **Dark Gray –** if entire record is dark gray, this indicates the end of all data areas for this page.

Differences with the Shortcut Menu in the Edit Panel

You learned details about the options available with the Edit Panel shortcut menu earlier in this course. With the Print Miner Module, there are a few differences. For example, the **Test** menu options do not appear. The following detail other differences.

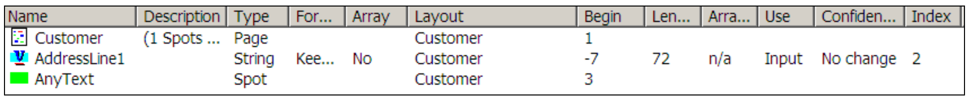
View Layout

View Layout Option



As with other data files, **View Layout** shows details about the mapped areas of the file in the Property Panel. There are a few differences, such as a new icon for a *spot* (the last listing in the illustration).

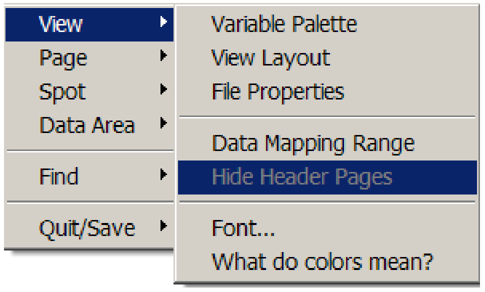
View Layout (Portion)



Name	Description	Type	For...	Array	Layout	Begin	Len...	Arra...	Use	Confiden...	Index
Customer	(1 Spots ...	Page			Customer	1					
AddressLine1		String	Kee...	No	Customer	-7	72	n/a	Input	No change	2
🟩 AnyText		Spot			Customer	3					

Hide Header Pages Option

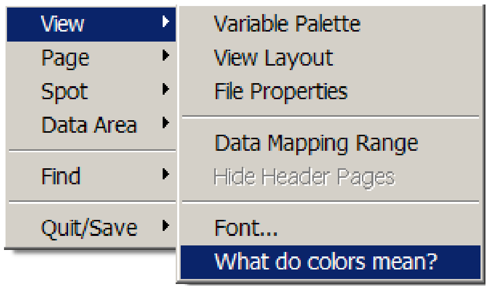
Hide Header Pages Option



The **Hide Header Pages** option is unique to Print Miner operation. You can select this to hide header pages so you do not need to deal with them while data mapping. This option will be dimmed and unavailable if you have not defined any header pages.

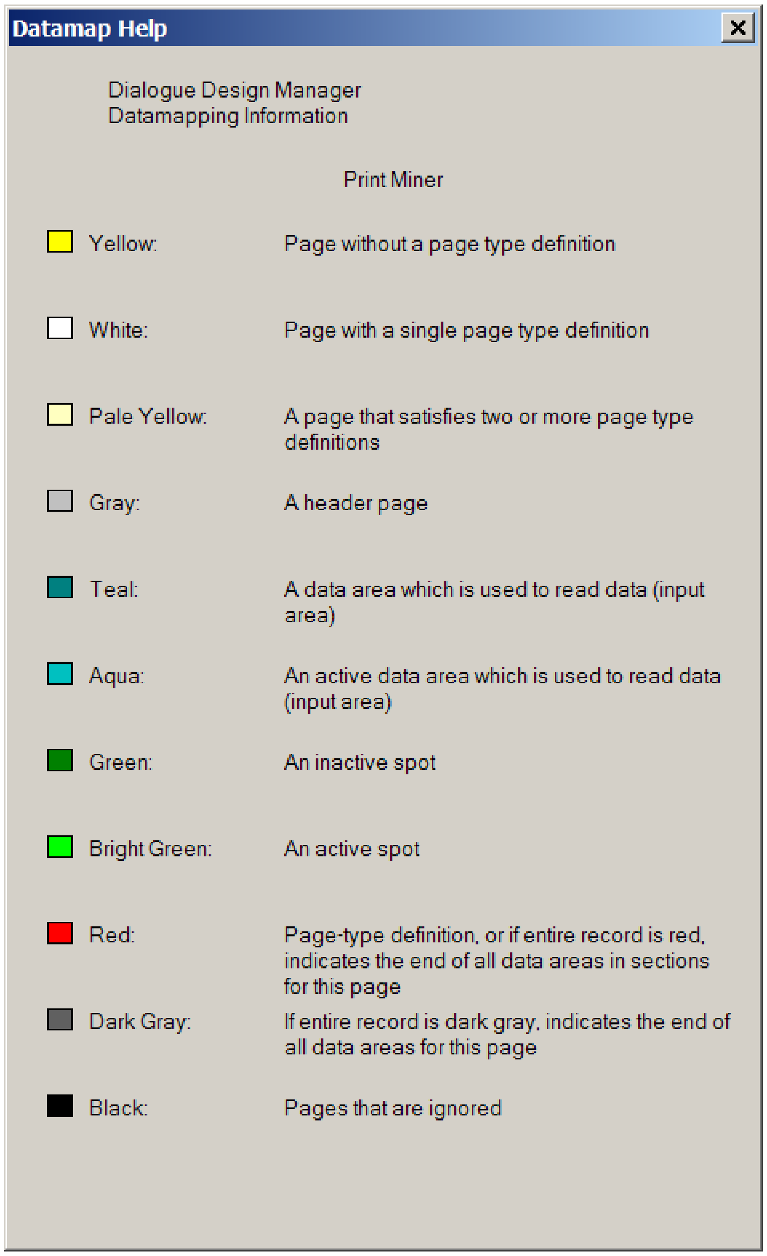
What do colors mean?

What do Colors Mean? Option



The Print Miner colors show module-specific information.

Colors in Print Miner


Datamap Help

Dialogue Design Manager
Datamapping Information

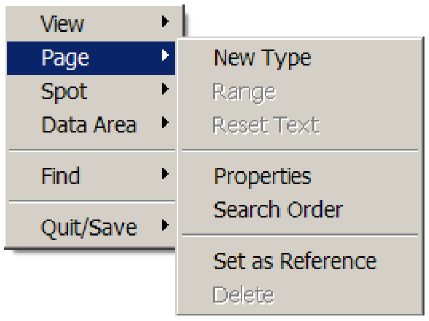
Print Miner

🟨	Yellow:	Page without a page type definition
⬜	White:	Page with a single page type definition
🟨	Pale Yellow:	A page that satisfies two or more page type definitions
⬜	Gray:	A header page
🟦	Teal:	A data area which is used to read data (input area)
🟦	Aqua:	An active data area which is used to read data (input area)
🟩	Green:	An inactive spot
🟩	Bright Green:	An active spot
🟥	Red:	Page-type definition, or if entire record is red, indicates the end of all data areas in sections for this page
⬛	Dark Gray:	If entire record is dark gray, indicates the end of all data areas for this page
⬛	Black:	Pages that are ignored

Page Menu

The **Page** options in the shortcut menu are unique to Print Miner.

Page Options in Shortcut Menu

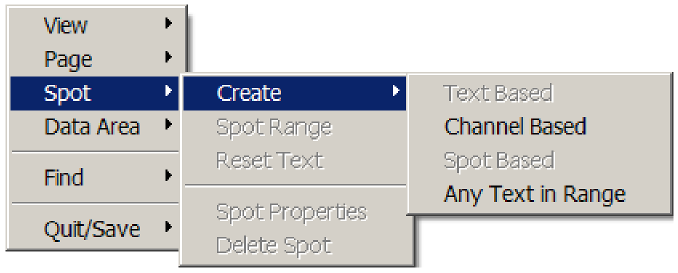


- **New Type** – create a new page **Type**.
- **Range** – set a **Range** of columns and records in which to search for the page type indicator that defines the page. If the page is already defined, this option will be dimmed.
- **Reset Text** – Redefines the text that is used to define the page.
- **Properties** – view the **Page Type Properties** of a page.
- **Search Order –** change the **Order** in which Dialogue searches for the page type. By default, this is in the order created.
- **Set as Reference –** set the page itself as a **Reference** point, rather than a spot or text area.
- **Delete – Delete** the active page **layout** and all references to it.

Spot

The **Spot** options in the shortcut menu are unique to Print Miner. The first-tier menu items will be discussed later in this chapter.

Spot Options in Shortcut Menu

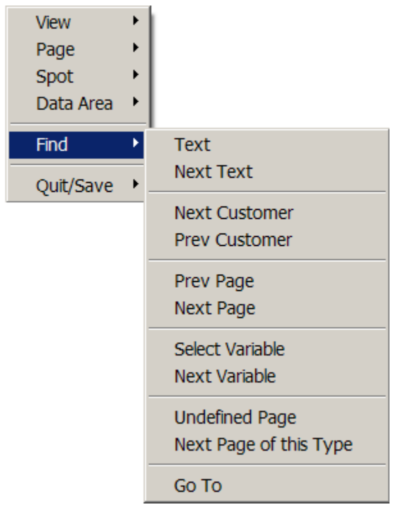


- **Create –** Define a new spot. From a submenu you specify if you will define the spot by text, channel, another spot, or any text within a range.
- **Spot Range –** set a range of columns and records in which to search for text that defines the spot.
- **Reset Text** – Reset text in which to search for text that defines the spot.
- **Spot Properties –** view the **Reference Spot Properties** of a spot.
- **Delete Spot –** Delete the active spot layout and all references to it.

Find

The **Find** menu options are longer than the standard set.

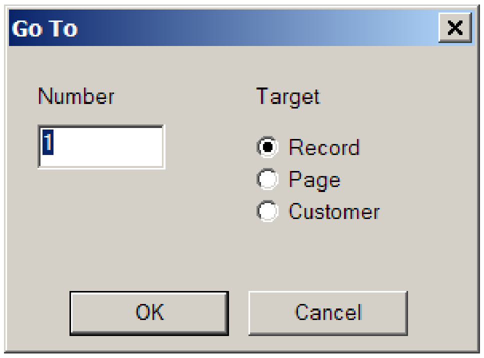
Find Options



- **Text** – to search for a string of text.
- **Next Text –** to search for the next occurrence of the highlighted text.
- **Next Customer –** to search for the next customer in the file.
- **Prev Customer –** to search for the previous customer in the file.
- **Prev Page –** to search for the previous page in the file.
- **Next Page –** to search for the next page in the file.
- **Select Variable –** to select a variable to find.
- **Next Variable** – to search for the next occurrence of the highlighted variable.
- **Undefined Page –** to search for the next undefined page in the file.
- **Next Page of this Type –** to search for the next page of the same type as the current page in the file.

- **Go To** – to access the **Go To** dialog box to find the **Nth** occurrence of a **Record**, **Page**, or **Customer**.

Go to Dialog Box



Map Customers in a Print Miner Data File

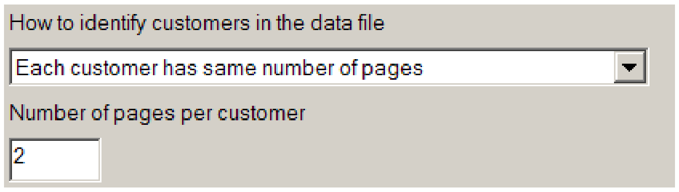
Decide How to Map a New Customer

The first thing to map depends on your choices when you defined the properties of the data file. It depends on whether you chose **Each customer has same number of pages** or **A new customer occurs on specified page type(s)**.

Each Customer has Same Number of Pages

- As an example, if each customer gets two pages, you chose **Each customer has the same number of pages** and set the **Number of pages per customer** to **2**. Dialogue knows that every other page starts a new customer.

Same Number of Pages


How to identify customers in the data file

Each customer has same number of pages

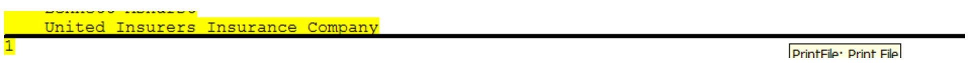
Number of pages per customer

2

- In addition, if you have indicated there are header pages to be ignored on the **Advanced** tab of the Property Panel, Dialogue will begin the count on the appropriate page.

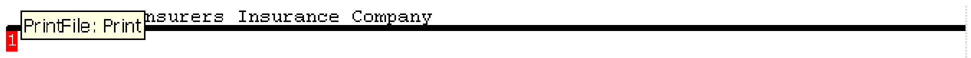
In this example, a **normal line** separates every page.

Normal Line



A **bold line** occurs after every two pages in your file (where a new customer starts).

Bold Line



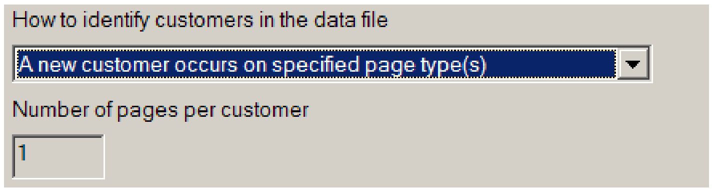
In addition, you have taken care of defining both pages for the customer, just by specifying the number of pages, as long as you do not have a reason to specify each **Page Type** as unique.

NOTE: **Page Types** will be discussed in the next section in this chapter.

A New Customer Occurs on Specified Page Type(s)

However, if the customers do *not* all have the same number of pages, you chose **A new customer occurs on specified page type(s)** in the **Basic** tab. The **Page Type** you should specify first when mapping the file is the one that starts the new customer.

Specified Page Type



How to identify customers in the data file

A new customer occurs on specified page type(s)

Number of pages per customer

1

Define Page Types

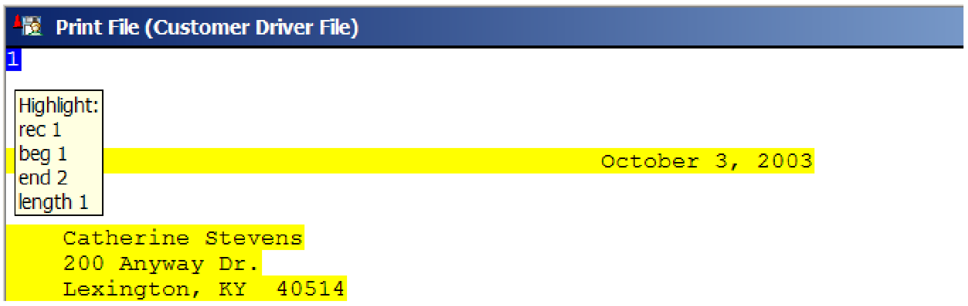
Before you define areas on pages, you define the **Page Type** for each page. You should define the **Page Type** for each new customer before any other page type, unless you have specified that **Each customer has the same number of pages**.

Identify an Area to Map

The first step in identifying **Page Types** is to find a consistent string of text on the same page, whenever it occurs. When you define the **Page Type**, all pages with the specified text string will be mapped using the same defined **Data Areas** and **Spots** as the original.

- If you use a **Page Type** for identifying a new customer, define the page that starts a customer first.
- **Highlight** the first area that you find consistently for each customer. Often this will be in the first line/channel) of a page.

Unmapped File in Edit Panel – A new Customer occurs on Specified Page Types

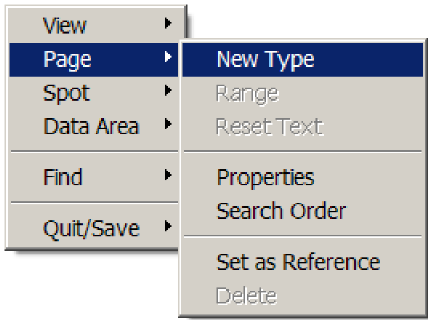


```
                                              October 3, 2003

    Catherine Stevens
    200 Anyway Dr.
    Lexington, KY  40514
```

Yellow highlighting, as in the above illustration, indicates a page has no page type definition. The pop-up only shows the position of the pointer. The first character in the file is a "1." It occurs at the beginning of each page. Therefore, it can be used to designate a new page.

- Right-click to access **Page**, then **New Type**.

Shortcut Menu

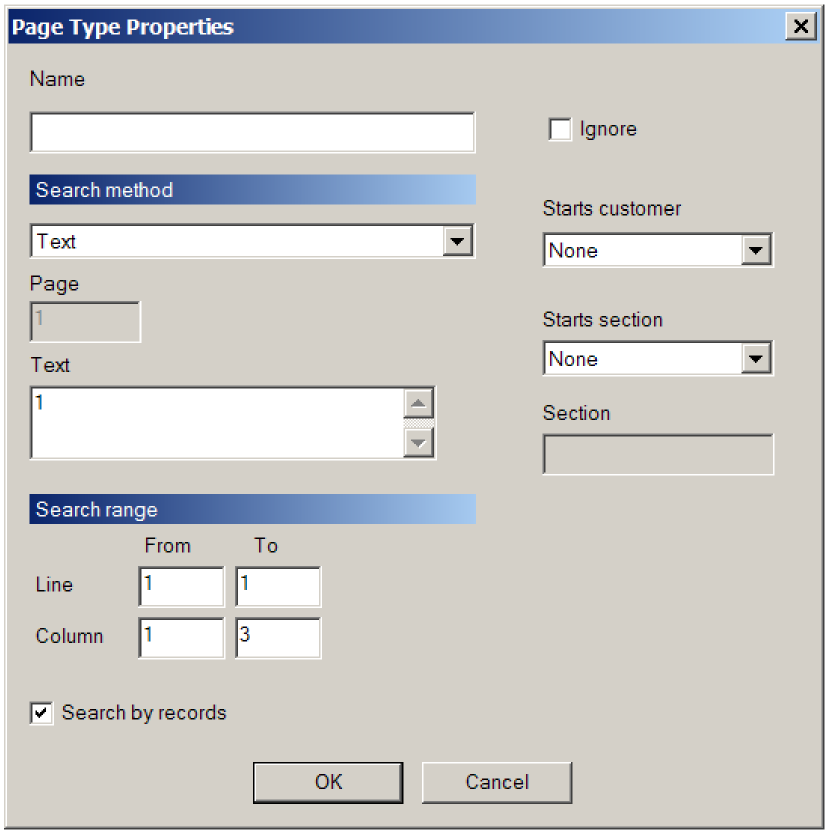


The **Page Type Properties** dialog box appears.

Page Type Properties Dialog Box

Here you can define the new **Page Type**.

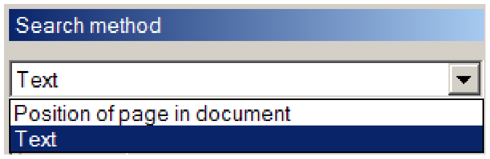
Page Type Properties Dialog Box



- Type a **Name** for your new Page Type.
  The name must be a unique identifier, much like a *Named Section* of data.

Search Method Area

Search Method Area

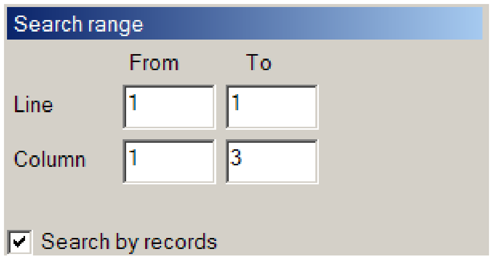


The **Search Method Area** defines how Dialogue will find this **Page Type**. You can search for:

- Position of Page in Document
- Specific Text

NOTE: Use the **Position of page in document** option if the text changes in the highlighted area, but the page occurs every *n* number of pages for each customer. This option would **not** be used to identify a new customer in this case, since the first page of a customer would start a new customer.

Search Range Area


Search range

	From	To
Line	1	1
Column	1	3

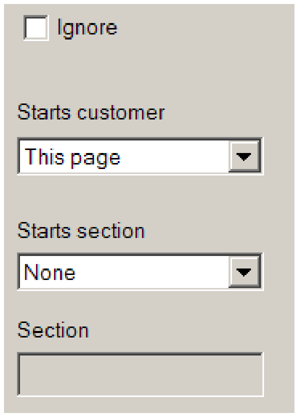
[✓] Search by records

With the **Search Range Area** you can search a range of lines and columns for the **Text**. This is helpful when the position of the text varies on each page.

- If you want to search specific lines, type the beginning **Line** in the **From** text box, and the ending **Line** in the **To** text box.
- If you want to search specific columns, type the beginning **Column** in the **From** text box, and the ending **Column** in the **To** text box.
- Select the **Records** check box if you want to search by records, rather than lines. Lines factor the carriage controls into the count, while records identify every record that is displayed on a separate line.

Remaining Areas

Remaining Areas


[ ] Ignore

Starts customer: This page

Starts section: None

Section:

- Select the **Ignore** check box if you want Dialogue to ignore this whole page. You would do this if the page were a page that did not contain any data you wanted to use, or "mine." ("Mine" is used in the context of mining data from a print file as you would mine gold from a gold mine.) This will suppress Engine messages that the page is undefined.
- Choose options in the **Starts Customer** drop-down list if:
  - **This Page** is the first page for all customers.

- **Next Page** if the current page is the last page for all customers. The reason you might use this is when the current page is the final page of the customer document and has unique wording like, "Total Amount Due." Sometimes recurring information happens on the last page of a customer document, and not the first.
- **None** is the default, and should be left in place if the page does not start a customer.

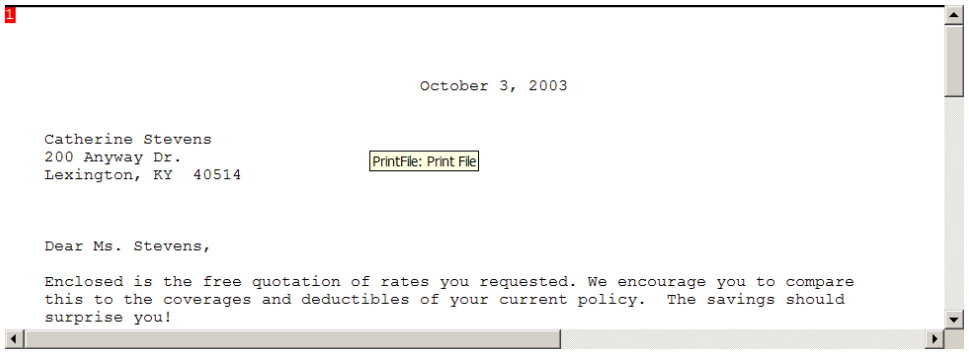
- Sections can be defined by Page Type by specifying options in the **Start Section** drop-down list:
  - **Always** means the Page Type always starts a new section of data for the customer, even when it recurs.
  - **First Occurrence** means the Page Type starts a new section of data for the customer, but only the first time it is found for the customer.
  - **None** is the default, and should be left in place if the page does not start a section.
- If you have chosen a **Section** option, type the name of the **Section** in the text box.
- Click **OK** to exit the dialog box.
- **Repeat** this procedure until all of the pages that contain data you want to use are defined.

TIP: You can identify **Page Types** by static text such as, "Page," "Dear," "Statement," "Account Number," "Date," and so on. Text can also be used to identify the start of a customer, when the text is something like "Page 1." Documents that contain OMR barcodes can be identified by the values in the code. Any unique portion of text can be utilized, such as the account number, when all account numbers begin with a prefix like, "ACC1000-," or any numbering method you use.

Mapped Data File Appearance

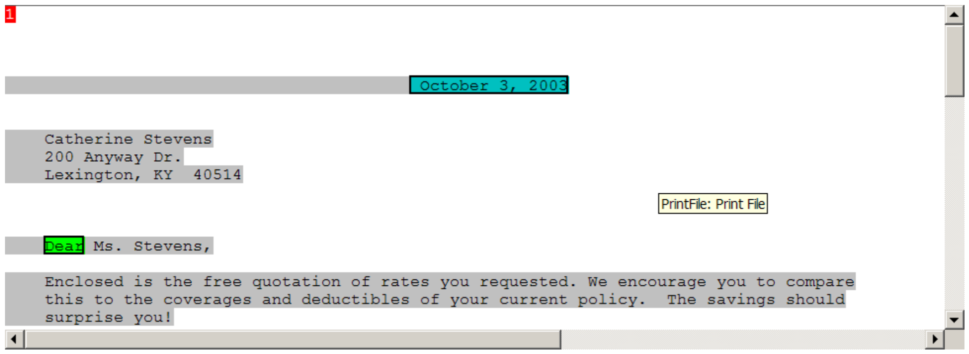
The text that was highlighted when you began to map the **Page Type** turns to red to signify that it is a page indicator. The text that was highlighted in yellow turns to white.

Mapped File


```
1

                                        October 3, 2003

    Catherine Stevens
    200 Anyway Dr.
    Lexington, KY  40514


    Dear Ms. Stevens,

    Enclosed is the free quotation of rates you requested. We encourage you to compare
    this to the coverages and deductibles of your current policy.  The savings should
    surprise you!
```

Header Pages and Pages with no Data

Header Page in Print Data File


```
1

                                        October 3, 2003

    Catherine Stevens
    200 Anyway Dr.
    Lexington, KY  40514


    Dear Ms. Stevens,

    Enclosed is the free quotation of rates you requested. We encourage you to compare
    this to the coverages and deductibles of your current policy.  The savings should
    surprise you!
```

The figure shows a header page, identified by the gray fields, that is ignored.

The "1" (red) in the first record in the illustration shows that this is a header page. It was mapped as the header page indicator because it does not appear in this position on any other kind of page. In the illustration, the pointer was placed on the second "1" at the start of the page. The pop-up shows that the "1" indicates a "Page: NewType." As mentioned, *page indicators* are highlighted in red.

NOTE: You do not have to set a **Page Type** on a page that contains no useable data. However, if you set "empty" pages to **Ignore** it will save processing time.

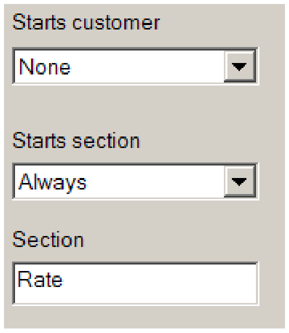
You may have specified a certain number of header pages in the Property Panel on the **Advanced** tab. If there is not always the same number of header pages in a file, define any header pages and select the **Ignore** check box in the **Page Type Properties** dialog box so Dialogue does not try to read these as customer records. For example, if a character occurs in the second column of the first line of all header pages, but otherwise it is blank, that character in that position can define the page as a header.

Pages that Start New Sections

Sections work the same way in Print Miner as they do in other data file types. Pages usually start sections, but so can spots.

You should define pages that will begin a new section of data, such as a new page or document. The name of the section page should be the same as the section name in the document or table you will use in Designer. One example of this is when you have a new account type on a table or page.

Page Type Properties - Section Properties


Starts customer

None

Starts section

Always

Section

Rate

Of course, if your data file only contains one page for each customer, you should be finished defining pages after the first type.

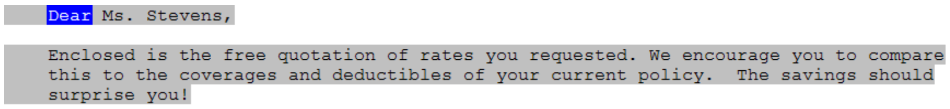
Define a Spot

Data can be mapped relative to a spot. A spot can float, thus enabling mapped data to float with it.

When defining a spot, the data can be mapped to find exact text, an exact position including a range of lines and columns (channels), or a position relative to any page or information that appears consistently in the file (a data area or another *spot*). This last method is called *a spot referencing a spot.*

You can drag your pointer across a range of text to select it for mapping a range. However, you are limited to 255 characters in a range to search for text. A spot can occur multiple times, anchor multiple other spots and data areas, but each spot must have a unique definition.

Data File in Edit Panel – Spot Highlighted

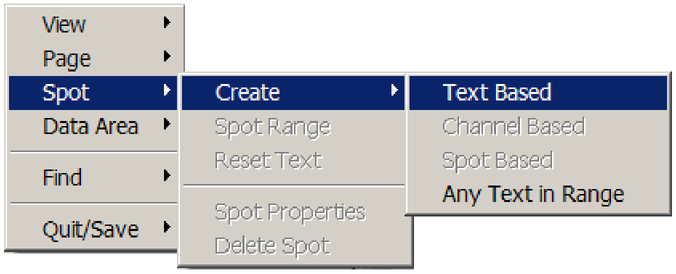

```
    Dear Ms. Stevens,

    Enclosed is the free quotation of rates you requested. We encourage you to compare
    this to the coverages and deductibles of your current policy.  The savings should
    surprise you!
```

In the preceding figure, a **spot** has been highlighted because the exact text occurs on each page in the same range of cells. It precedes information that will be used from the data file, so you will want to map it as a spot that is used to define a data area.

To define a spot:

- Highlight the area for the spot. This activates the area.
- **TIP:** Areas where text **never** occurs in the file make good starting and ending places for the **Search Range** for spots and data areas.
- Right-click to access the shortcut menu.
- Select **Spot**, **Create**, **Text Based**. Alternately, you could press CTRL + T.

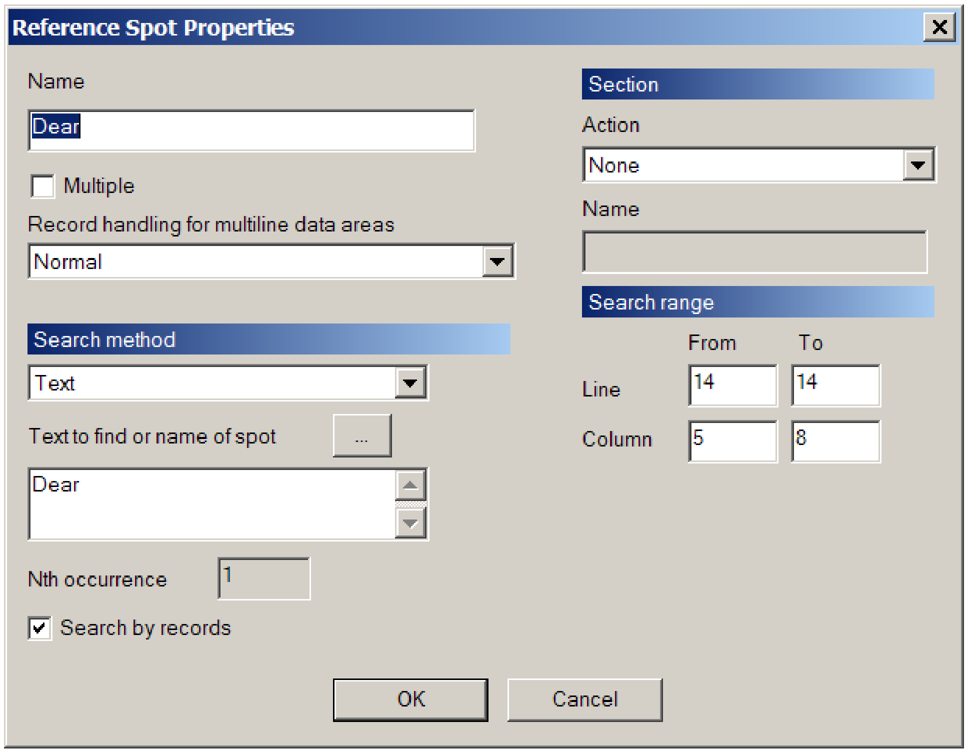
Create Spot Menus



The **Reference Spot Properties** dialog box appears. The reason you choose **Text Based** in this situation is that the text will always be the same: *Dear*.

Defining Spot Properties

Reference Spot Properties Dialog Box



Reference Spot Properties

Name: Dear

☐ Multiple

Record handling for multiline data areas: Normal

Search method: Text

Text to find or name of spot: Dear

Nth occurrence: 1

☑ Search by records

Section

Action: None

Name:

Search range

	From	To
Line	14	14
Column	5	8

Name

Dialogue will automatically **Name** the spot for you based on the text that you have highlighted, but you can change it here.

Multiple Check Box

Select the **Multiple** check box if the spot will occur multiple times on the page.

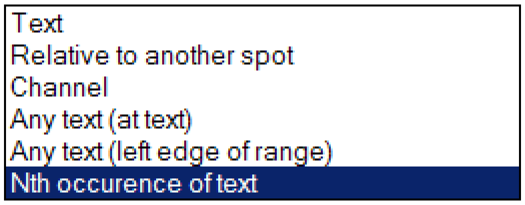
The multiple check box will look for multiple occurrences of the spot in the **Search Range**. If you do not make this selection, only the first occurrence of the spot in the range is found.

When the spot is specified and found multiple times, all data relative to the spot is mapped multiple times. In other words, relationships to the spot exist for all occurrences of the spot.

Search Method

Under **Search Method**, choose one of the following ways for Dialogue to search for text from the drop-down list:

Search Method Options


Text
Relative to another spot
Channel
Any text (at text)
Any text (left edge of range)
Nth occurence of text

- **Text** will already be highlighted if you chose **Text Based** on the shortcut menu.
    - If you choose **Text Based** when you were on the shortcut menu, and text is already highlighted in the data file, Dialogue will place the text in the **Text to find or name of spot** text box automatically. **Text** options will work more effectively if you highlight the space before and after the word, so that you will not get larger words of which your word is a part. For instance, if you highlight "(space)Dear(space)" you will not get "Dearest".
    - You can also change or add text in the **Text to find or name of spot** text box, but the range must include the area where the additional or changed text is found.
- **Relative to another spot** will already be highlighted if you chose **Spot Based** on the shortcut menu. **Relative to another spot** will make your current spot relative to a spot already defined in the file.
    - Click the ... button to get a list of the spots already defined in the data file.

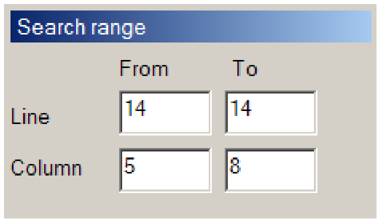
This feature is useful for determining the start of a section. Sometimes unique data that can signal the start of a section occurs before or after the actual start of the section data. Specify this unique data as a spot and specify the section spot relative to the first spot.

- **Channel** will already be highlighted if you chose **Channel Based** on the shortcut menu. Use **Channel** to search for the character or text in a certain channel (column) of the file. You could also use **Text** as your choice here to define the channels, but identifying the spot as **Channel Based** speeds the search by searching in a specific channel.
- **Any text (at text)** to search for any character or text starting at the beginning of the text. A range is specified for the spot and is set where any text is found in the range. The position for relative data is calculated based on the leftmost position of text in the area.
- **Any text (left edge of range)** to search for the character or text starting at the left of the range as it is specified. A range is specified for the spot and is set

when any text is found in the range. The position for relative data is calculated based on the leftmost position in the range, whether any text exists or not.

- **Nth Occurrence of text** to search for the character or text at a specified number of its occurrence. For example, "Dear" occurs 3 times on the page within the range you have chosen and you want the second occurrence.
  - If you make this choice, type the number of the occurrence you want to specify in the **Nth** text box.
- If you chose **Any text in Range** on the shortcut menu, you also specify a starting and ending range.

Search Range Area



	From	To
Line	14	14
Column	5	8

- In the **Search Range Area**, define a range by using the **Line:** and **Column:** text boxes.

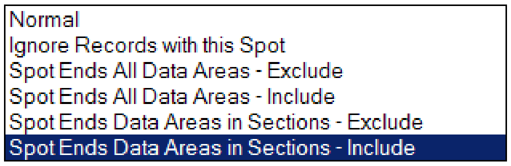
These ranges will already be filled with the numbers of the range you highlighted in the data file, but you have an opportunity to change them here.

- If you want to define the range in records instead of lines, select the **Search by records** check box. This is the default setting.
- Click **OK** to conclude defining the spot properties and return to the data file.

NOTE: When counting for relative specifications, begin counting from the upper left of the relative spot, starting with 0. When making a spot relative to another spot, be sure the spot you are defining is highlighted. You can click an empty spot on the page to deactivate a spot.

Record Handling for Multiline Data Area

Multiline Options


Normal
Ignore Records with this Spot
Spot Ends All Data Areas - Exclude
Spot Ends All Data Areas - Include
Spot Ends Data Areas in Sections - Exclude
Spot Ends Data Areas in Sections - Include

- **Normal** is the default and lets this spot act as any spot would.

- **Ignore Records with this Spot** – the record is ignored. You would use this when it contains a total that you want to ignore when converting multiline data areas into arrays.
- **Spot Ends** All Data Areas – the record ends all data areas above it. You would use this in setting an end to all columns of a table at once, so that you do not have to set an end to each column. It also allows you to extend all columns to the bottom of the page and end them by this spot.
  - Choose to **Exclude** or **Include** the current spot.
- **Spot Ends All Data Areas In Sections** means that the record ends the processing of the data area within the current section.
  - Choose to **Exclude** or **Include** the current spot.

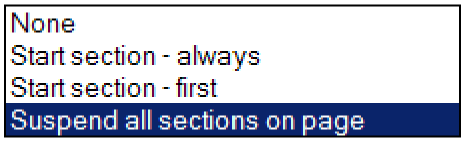
NOTE: When a spot has its **Record Handling for Multiline Data Areas** properties set to **Ignore Records with this Spot**, the record highlighting turns black.

Section Area

A section with a given name can be set to start on a **spot**. A spot can also be specified on a page that indicates that section data does not go beyond this spot. This allows non-section data to be read from the bottom of a page that starts a section and ensures that the ends of sections on the page are recognized.

NOTE: When the Engine runs, these spot properties help Dialogue process section-based information like sections in other data files.

Section Options



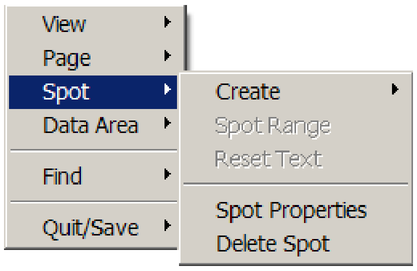
- **None** designates that the spot does not control section data.
- **Start Section – always** designates that the spot always starts a new section. The text box below the drop-down list becomes available for you to identify the section name.
- **Start Section – first** designates that the spot starts a section on its first occurrence. The text box below the drop-down list becomes available for you to identify the section name.
- **Suspend all sections on page** designates that the spot stops all section-based data on the page. This is used primarily when section data is identified in a page, but space at the bottom of the page is reserved for page footers. This

option halts the section processing so that the footers are ignored. The section processing resumes on the next page.

Other Spot Options

Other options on the shortcut menu that appear after selecting **Spot** include:

Submenu Options for Spot



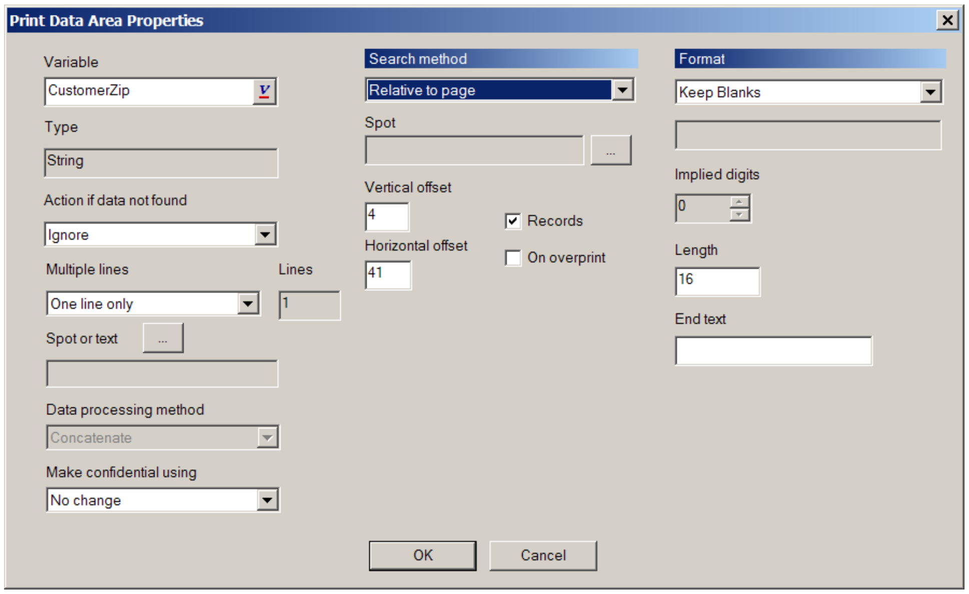
Besides being able to **Create** the spot, and view **Spot Properties**, you can select **Spot Range**, **Reset the Text**, and **Delete the Spot**.

- To use **Spot Range**, highlight a new area that includes the spot and choose this option. Dialogue will ask if you want to include **Multiple** occurrences of the text.
- **Reset Text** lets you change the text that controls the spot.
- **Delete Spot** lets you totally delete the spot.

Multiple Lines

You can find additional spot settings on the **Print Data Area Properties** dialog box. Right-click the spot and select **Data area** -> **Properties** from the shortcut menu to access it.

Print Data Area Properties dialog box



Variable: CustomerZip
Type: String
Action if data not found: Ignore
Multiple lines: One line only
Lines: 1
Spot or text:
Data processing method: Concatenate
Make confidential using: No change
Search method: Relative to page
Spot:
Vertical offset: 4
Horizontal offset: 41
[✓] Records
[ ] On overprint
Format: Keep Blanks
Implied digits: 0
Length: 16
End text:

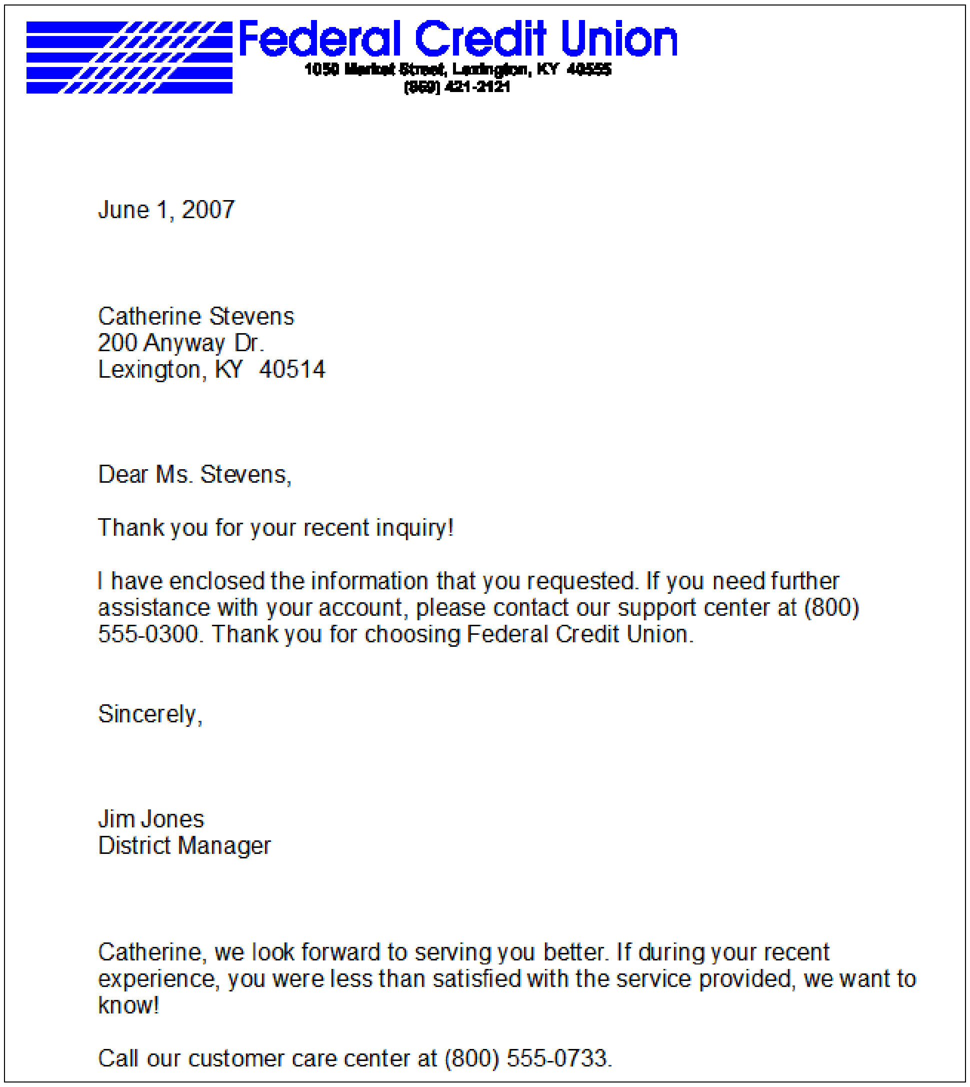
Use the **Multiple lines** area to specify whether data can span more than one line, and if so, how Dialogue finds the end of the spot data.
Spot options are:

- **Until specific spot** - Data continues until the spot specified in the Spot or text box is found or the number of lines in the Lines box is reached (not including the spot text). Click the button to open the Select the Spot dialog box. Select a spot defined in the data file and click OK.
- **Until any spot** - Data continues until any spot is found or the number of lines in the Lines box is reached. Data does not include the specified spot.
- **Until next spot** - Data continues until a spot is found after the first spot.

7.7 Exercise: Print Miner

In this exercise you create Federal Credit Union's inquiry letter using a print file as a Customer Driver File.

Composed Page



Federal Credit Union
1050 Market Street, Lexington, KY 40555
(859) 421-2121

June 1, 2007

Catherine Stevens
200 Anyway Dr.
Lexington, KY 40514

Dear Ms. Stevens,

Thank you for your recent inquiry!

I have enclosed the information that you requested. If you need further assistance with your account, please contact our support center at (800) 555-0300. Thank you for choosing Federal Credit Union.

Sincerely,

Jim Jones
District Manager

Catherine, we look forward to serving you better. If during your recent experience, you were less than satisfied with the service provided, we want to know!

Call our customer care center at (800) 555-0733.

Create a Data File for Print Miner

1. In Design Manager, expand the **Exercise 07 Print Miner** folder.
2. Right-click the **Data Files** heading in this folder.
3. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
4. Enter `Print File` in the **Name** field.
5. Choose **Customer driver file** from the **File type** drop-down list.
6. Choose **Print file** from the **File Format** drop-down list.
7. Click **Finish**.
   The **Data File** appears in the Property Panel for you to define.
8. Click the **Basic** tab.
9. In the **File format** area, choose **Line data** from the **Print data type** drop-down list.
   You will leave the default settings for the remaining properties on the **Basic** tab and for the **Advanced** tab properties.
10. Click the **Test Data Source** tab.
11. In the **File to use for data mapping** field enter:
    `C:\210 Advanced Data Concepts\Data Files\Print Miner.txt`
12. Click the **Production Data Source** tab.
13. In the **File to use in production** field, enter: `EXprint`.
14. **Save** (but do not exit) the Property Panel.

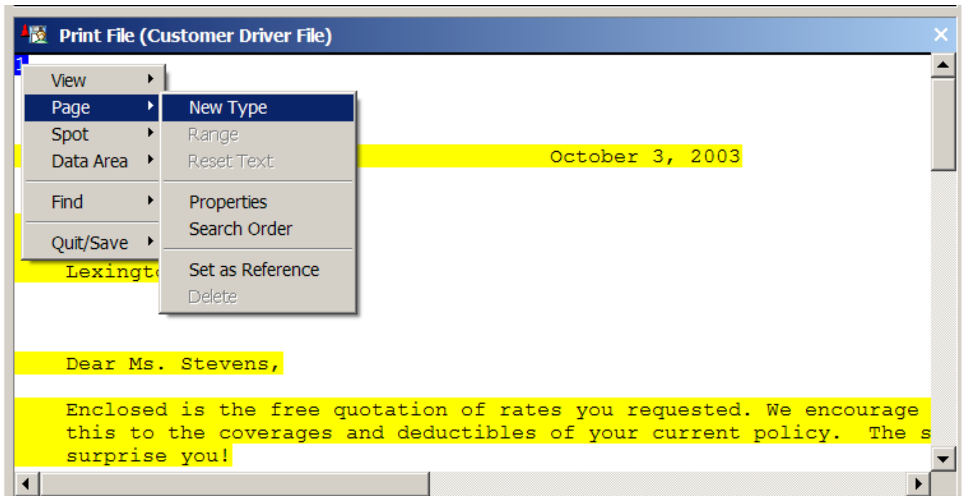
Map the Print File

1. Drag the **Print File** from the Property Panel to the Edit Panel.

Dialogue highlights the contents of the print file in yellow. This indicates you have yet to define the page type.

2. Open the **Variable Panel** and browse to the **Exercise 07 Print Miner** folder.
3. Highlight the first data area in the file in the upper left (the number “1”).
4. Right-click and choose **Page** from the shortcut menu.
5. Choose **New Type** from the submenu.

Select Page/New Type

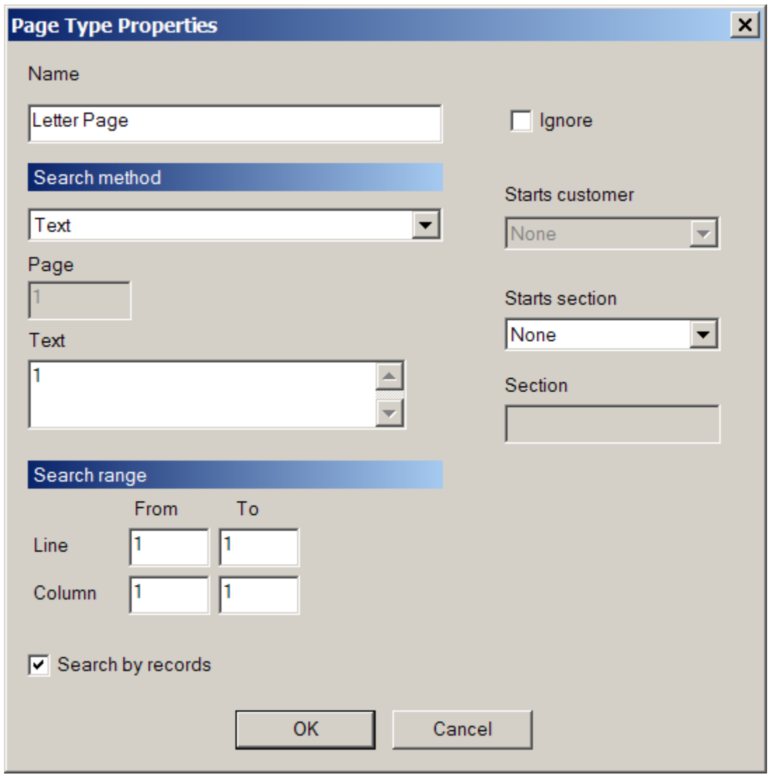


```
                                      October 3, 2003

Lexingt

Dear Ms. Stevens,

Enclosed is the free quotation of rates you requested. We encourage
this to the coverages and deductibles of your current policy.  The s
surprise you!
```

The **Page Type Properties** dialog box appears.

6. In the **Name** text box, enter `Letter Page`.

Page Type Properties


Page Type Properties

Name: Letter Page

☐ Ignore

Search method: Text

Page: 1

Text: 1

Starts customer: None

Starts section: None

Section:

Search range

	From	To
Line	1	1
Column	1	1

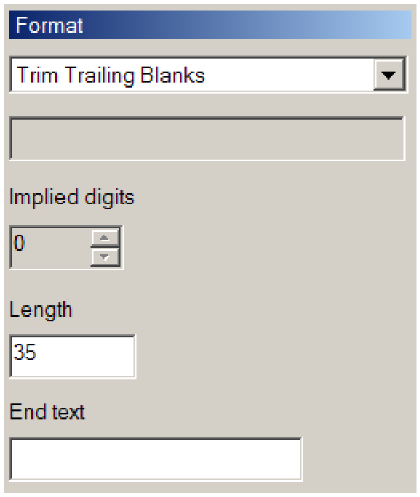
☑ Search by records

7. Click **OK**.

Dialogue gives the Page Type area the color red and removes the yellow highlight from the remainder of the file.

8. Highlight Catherine Stevens.
9. Double-click the **CustomerName** variable from Variable Panel.
   The Print Data Area Properties dialog box appears.
10. Choose **Trim Trailing Blanks** from the **Format** list.
11. Type 35 in the **Length** text box.
12. Click **OK**.
    The data area in the Edit Panel turns teal in color.

Print Data Area Properties


Format
Trim Trailing Blanks
Implied digits
0
Length
35
End text

13. Repeat steps 8 through 12 to map the **CustomerAddress1** and the **CustomerAddress2** variables to the next two lines in the print file.
14. Save and close the Edit Panel.

Package and Run the Application

1. Drag the **Print File** you created in this exercise to the **Print Miner Application**.

NOTE: Ensure the order of the data files, from top to bottom, is as follows: Initialization File, Print File, and Reference File.

2. Package the Print Miner Application.
3. View the results in Exstream Viewer. The print file supplied the data for the address blocks on the two letters.

7.8 Exercise: Create a Spot

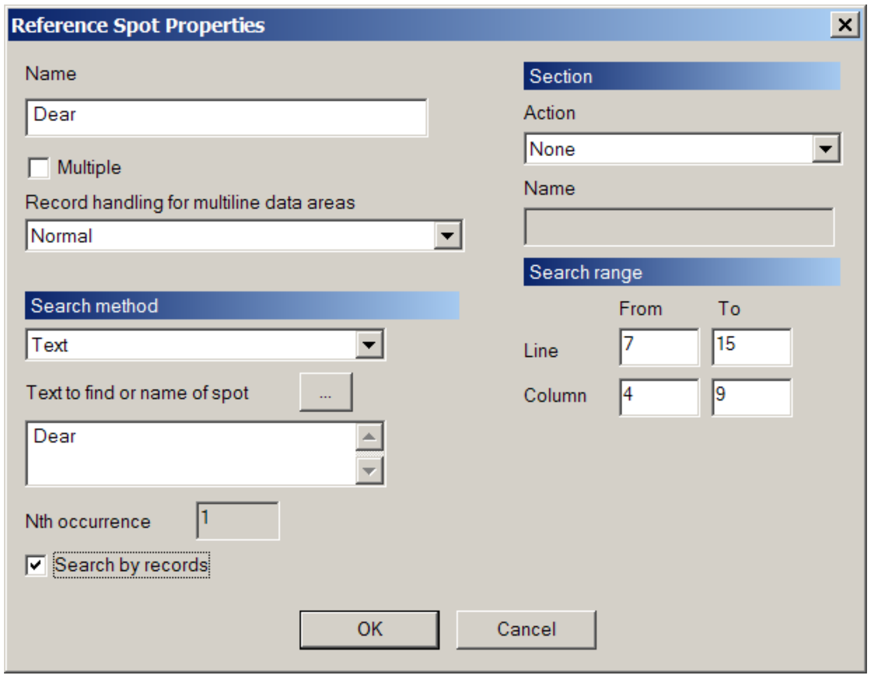
Create a Spot

1. Drag the **Print File** to the Edit Panel.
2. Scroll in the Edit Panel to see that the location of the salutation is a little different on the two letters in the file. In the first letter the greeting appears on the *fourth* line below the address block. In the second letter the greeting appears on the *second* line below the address block. Common to both is the word *Dear*. You will create a spot based on this word.
3. On the first page highlight the word **Dear**. Include the space before and the space after.
4. Right-click, choose **Spot**, choose **Create**, and select **Text Based**.

The **Reference Spot Properties** dialog box appears. To reduce keystrokes, the word *Dear* already appears as the **Name** of this spot.

5. Change the **From/To** number range for **Line** to read **7** to **15**. The Engine will search these lines for the text *Dear* on each page.

Reference Spot Properties



Name: Dear
Multiple: unchecked
Record handling for multiline data areas: Normal
Search method: Text
Text to find or name of spot: Dear
Nth occurrence: 1
[✓] Search by records
Action: None
Name:

	From	To
Line	7	15
Column	4	9

6. Leave the remaining properties at their default settings. Click **OK**.

Dialogue gives this data area a bright green color to indicate it is a spot.

Map the CustomerGreeting Variable

1. Highlight **Ms. Stevens** and the comma.
2. Double-click the **CustomerGreeting** variable from the Variable Panel.

The **Print Data Area Properties** dialog box appears. It already identifies the **Dear** spot in the **Search Method** area.

3. In the **Format** drop-down list (upper right) choose **Trim Trailing Blanks**.
4. Type 35 in the **Length** text box.
5. Leave the other properties at their default values in the **Print Data Area Properties** dialog box. Click **OK**.
   You return to the Edit Panel with the spot data mapped.
6. In the Edit Panel, scroll to the second letter and see how your mapping looks at the top of that page.
7. Save and close the Edit Panel.

Add Variable to Design Page

1. In the Library, expand the Print Miner Document under the Print Driver Application.
2. Drag the Letter Page to the Edit Panel to open it in Designer.
3. The CustomerName variable appears three times on the design page. Replace the second and third occurrences with the CustomerGreeting variable that you just mapped.
4. Save the page and close Designer.

Package and Run the Engine

1. Package and run the Print Miner Application.
2. Review the output in the Exstream Viewer.
   - The Engine used a *spot* to locate and read each greeting in the print file.
   - The **CustomerGreeting** variable you placed in the text wrote the string that the Engine read as the greeting to each customer.

Chapter 8: PDF Form Miner

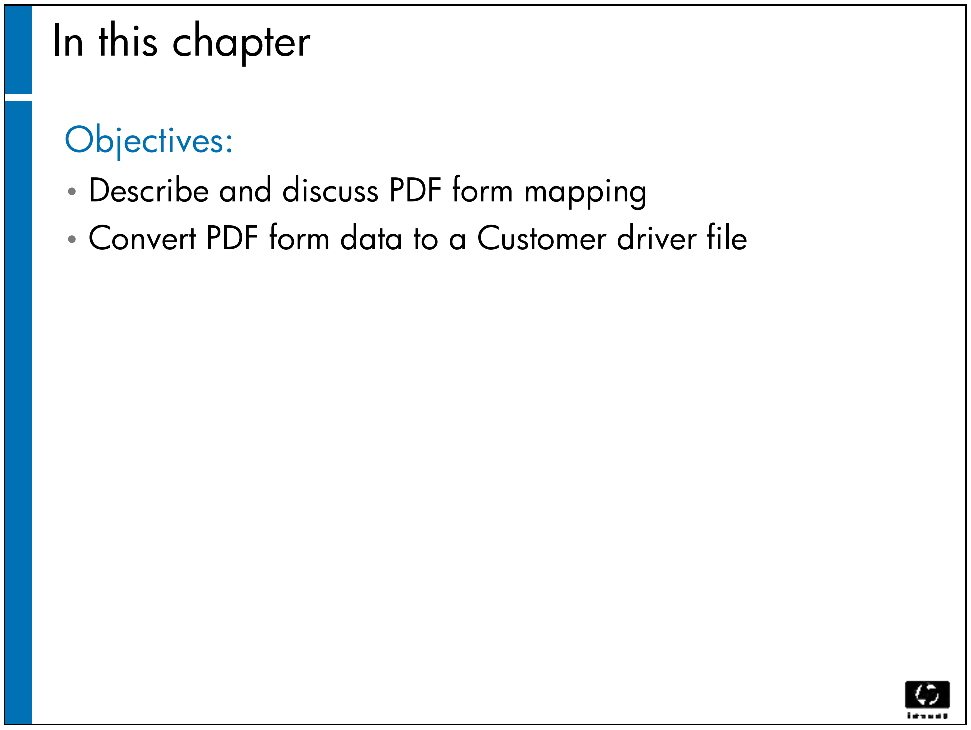
8.1 In this Chapter

The PDF Form Miner module enables you to map data from pre-existing PDF forms into Dialogue variables. You can use a PDF form to generate a file or update a system. You can also convert the PDF form data to a Customer driver, Reference, or Initialization file.

Objectives

By the end of this chapter you will:

- Describe and discuss PDF form mapping.
- Convert PDF form data to a Customer driver file.



In this chapter

Objectives:

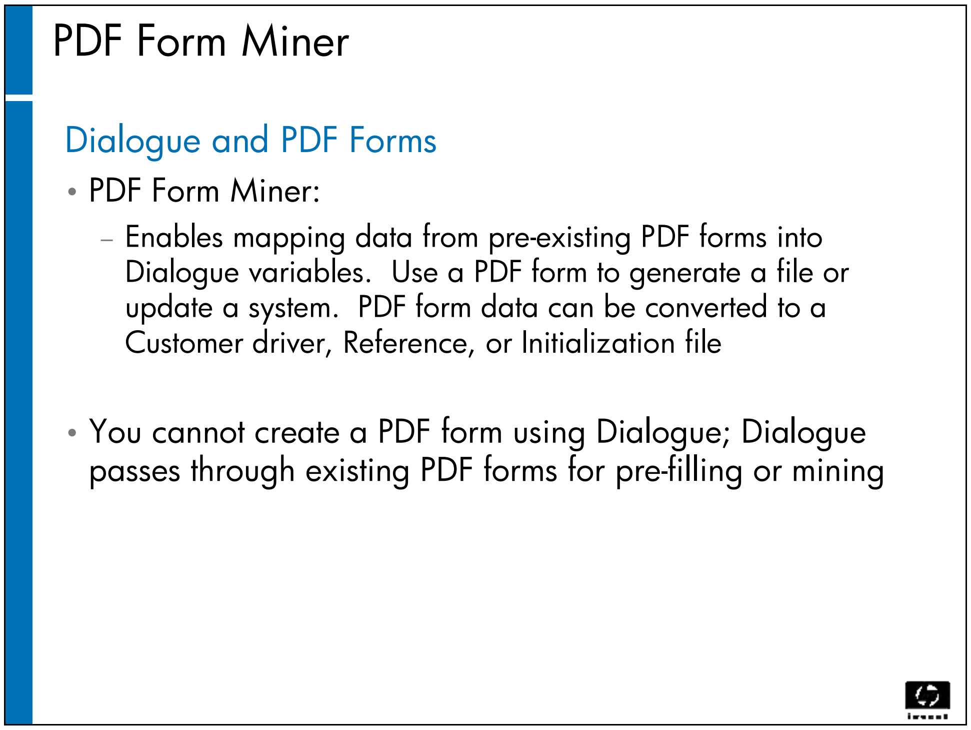
- Describe and discuss PDF form mapping
- Convert PDF form data to a Customer driver file

For more information

For more information, refer to the following Dialogue documentation:

- Populating and Mining PDF Forms (PDFForm.pdf).

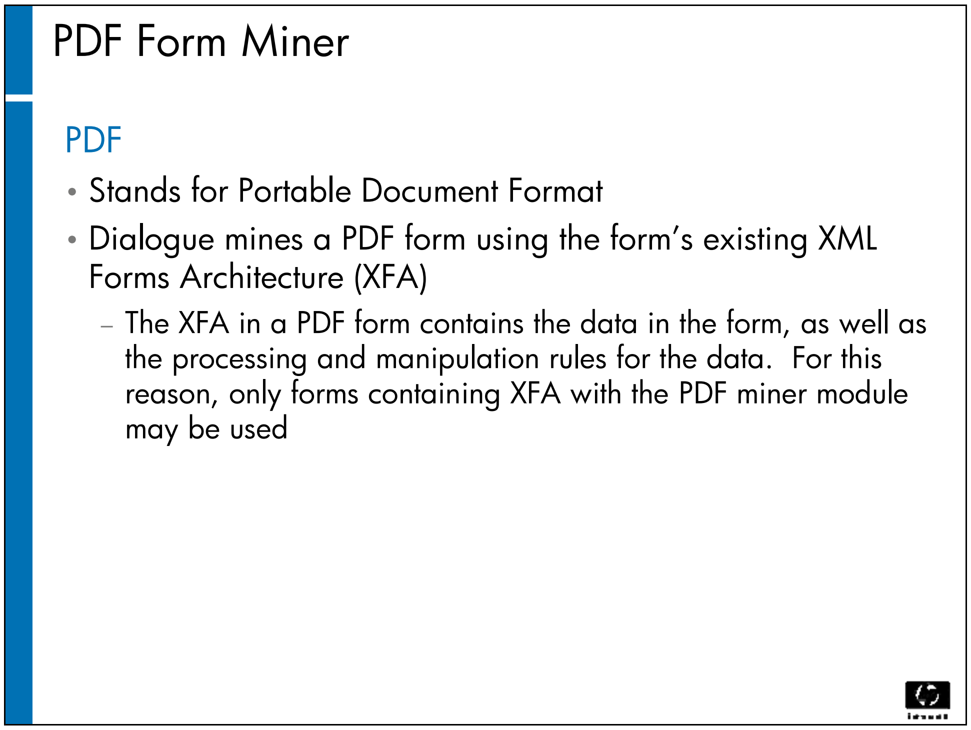
8.2 PDF Form Miner Overview



PDF Form Miner

Dialogue and PDF Forms

- PDF Form Miner:
  - Enables mapping data from pre-existing PDF forms into Dialogue variables. Use a PDF form to generate a file or update a system. PDF form data can be converted to a Customer driver, Reference, or Initialization file
- You cannot create a PDF form using Dialogue; Dialogue passes through existing PDF forms for pre-filling or mining



PDF Form Miner

PDF

- Stands for Portable Document Format
- Dialogue mines a PDF form using the form's existing XML Forms Architecture (XFA)
  - The XFA in a PDF form contains the data in the form, as well as the processing and manipulation rules for the data. For this reason, only forms containing XFA with the PDF miner module may be used

You can mine data from multiple PDF forms by using either a:

- **Placeholder variable** - Specified PDF forms are read during processing (based on the timing of the data file). The Dialogue Engine extracts the XML data and passes it to Dialogue variables. Using Placeholder variables enables you to mine multiple forms during processing.
- **Connector (if you have purchased the Dynamic Data Access module)** -The DDA routine extracts data headers from the underlying XML layer of external PDF forms. The DDA routine then passes the data to Dialogue variables. Using a DDA routine enables you to mine PDF forms for data in real-time.

NOTE: PDF forms must contain XML-based XFA to be mined. The XFA must be version 2.2.

To verify that the **PDF form miner** module is enabled on your system, drag **System Settings** to the Property Panel. Click the **Key** tab, the PDF form miner module appears under the **Advanced data access** area. If its check box is selected, you have access to the features in this chapter.

8.3 Exercise: Convert PDF form Data to a Customer Driver File

The PDF form miner enables you to convert to many data file types. These data file types include:

- Customer driver
- Initialization
- Reference
- Report
- Post-sort initialization
- Post-sort report
- Auxiliary layout

A. Create a Customer Driver File

In this section you will create a Customer driver file you will map in a later exercise.

1. Right-click the Data Files heading under the Exercise 08 PDF Form Miner folder.
2. Choose New Data File from the shortcut menu.

The **New Data File** dialog box appears.

3. Type `PDF Customer Driver File` in the **Name** text box.
4. In the **File type** drop-down list, choose **Customer driver file**.
5. In the **File format** drop-down list, choose **PDF Form**.
6. Click **Finish**.
   The file appears in the Property Panel for you to define.

Basic Tab

With a **File Format** of PDF Form, additional properties appear inside the **File Format** area on the **Basic** tab. Before you define the properties in the **File Format** area, you must decide the mapping method to use.

7. Click the **Basic** tab.
8. From the **Data mapping method** drop-down list, choose **Manual**.
9. Click the **Test Data Source** tab.

Data Source Tabs

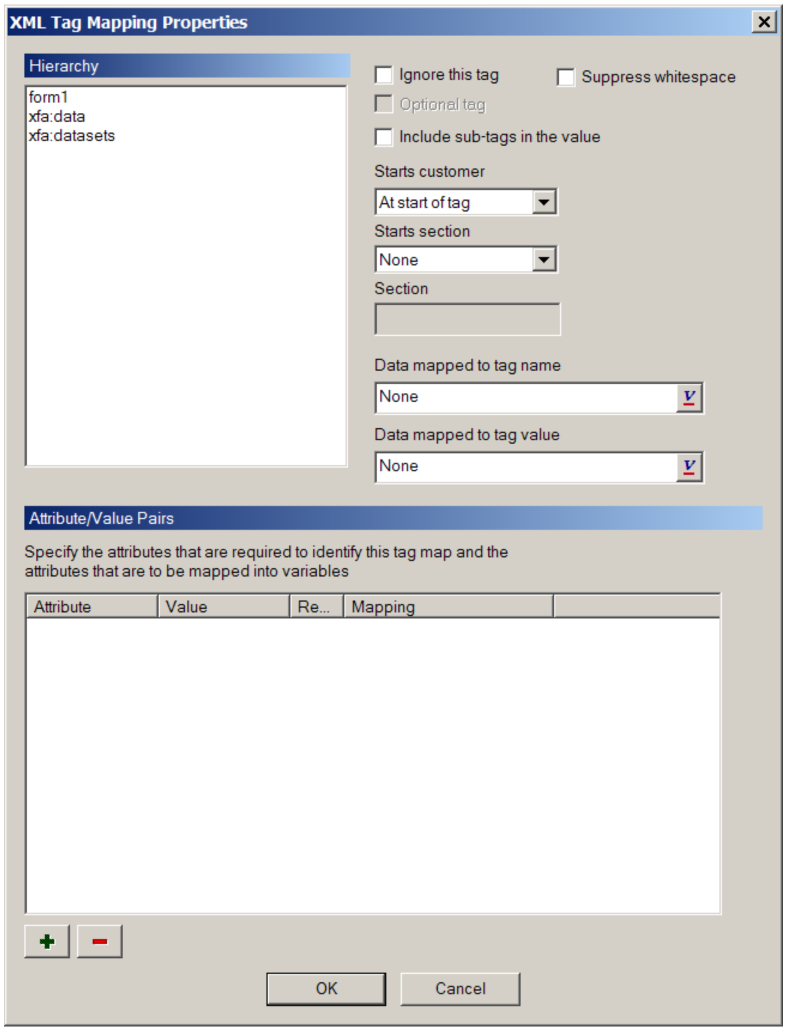
The properties on the **Test Data Source** tab pertain to test runs of the application.

10. From the **Type** drop-down list, select **File**.
11. Select the button and browse to the `C:\210 Advanced Data Concepts\Data Files\Customer-Info.pdf` PDF form.
12. Click the **Production Data Source** tab.
13. Enter `DD:CUSTINFO` in the **File to use in production** box.
14. Save and close the Property Panel.

B. Map the Customer Driver File

1. Drag the `PDF Customer Driver File` to the Edit Panel.
2. Right-click any empty area in the Edit Panel.
3. Select **Tag** -> **Auto-Map** from the shortcut menu.
   The **XML Auto-Map** dialog box opens, for you to complete the properties.
4. Click the button to browse to the **Exercise 08 PDF Form Miner** folder.
5. Click **OK**.
6. Click **OK** on the **XML Auto-Map** dialog box.
7. Highlight the `form1` tag to define the beginning of customer information.
8. Right-click and select **Tag** -> **Properties** from the shortcut menu.
   The **XML Tag Mapping Properties** dialog box opens.

XML Tag Mapping Properties dialog box



9. Select At start of tag from the Starts customer drop-down list.
10. Click OK.
11. Save and close the Edit Panel.

C. Add Variables to Design Page

1. Drag the PDF Customer Driver File to the PDF Form Miner Application in the Exercise 08 PDF Form Miner folder.

NOTE: Ensure the order of the data files, from top to bottom, is as follows: Initialization File, PDF Customer Driver, and Reference File.

2. In the Library, expand the PDF Print Form Document under the PDF Form Miner Application.
3. Drag the Letter Page to the Edit Panel to open it in Designer.

4. Click the V button to open the Variable panel.
5. In the letter, highlight Name in the Address Block.
6. Double-click the Name variable from Variable Panel in the Exercise 08 PDF Form Miner folder.
7. Repeat steps 5 and 6 twice to include all occurrences of the Name in the file.
8. Map the following text to the corresponding variables:

Data Area	**Variable to map**
Address	**Address**
City	**City**
State	**State**
Country	**Country**
ZipCode	**ZipCode**

9. Save the page and close Designer.

D. Package and Run the Engine

1. Package and run the PDF Form Miner Application.
2. Review the output in the Exstream Viewer.

The Engine used the variables you added to create the output.

Chapter 9: ODBC Access Data Files

9.1 In this Chapter

With the optional ODBC Access module, the Engine can read data directly from databases, instead of XML, PDF Form Miner, Print Mine, or flat files. This chapter contains background concepts, basic terminology, and a detailed discussion on how to create data files for this module.

Objectives

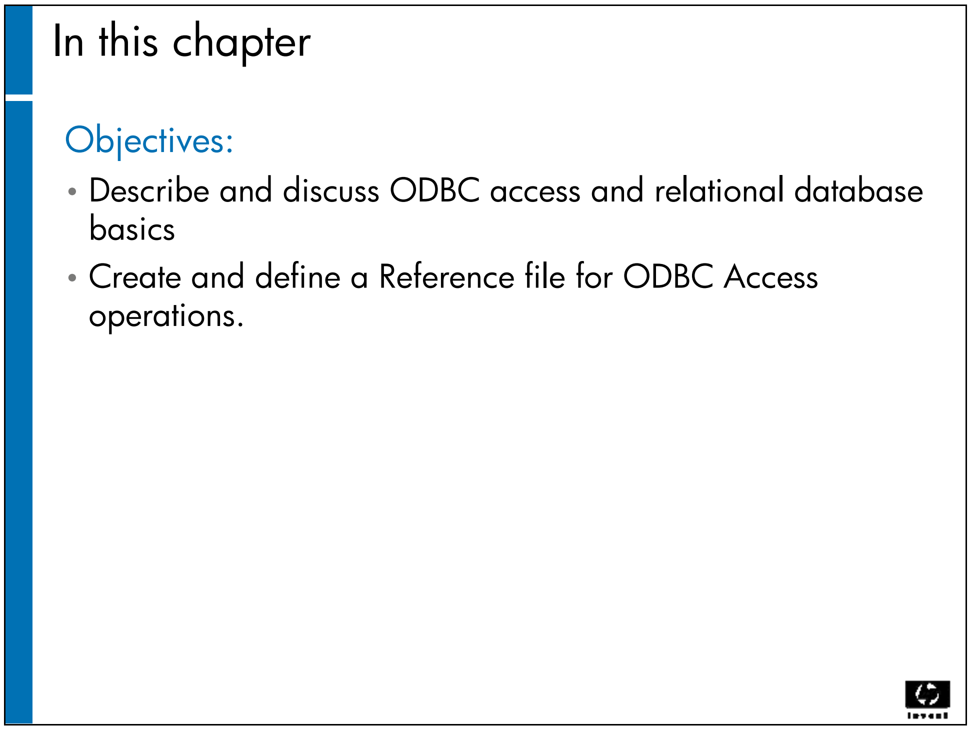
By the end of this chapter you will:

- Describe and discuss ODBC Access and relational database basics.
- Create and define a Customer Driver File for ODBC Access operations.
- Create and define a Reference File for ODBC Access operations.

For more information

For more information, refer to the following Dialogue documentation:

- ODBC Access guide (ODBC.pdf).

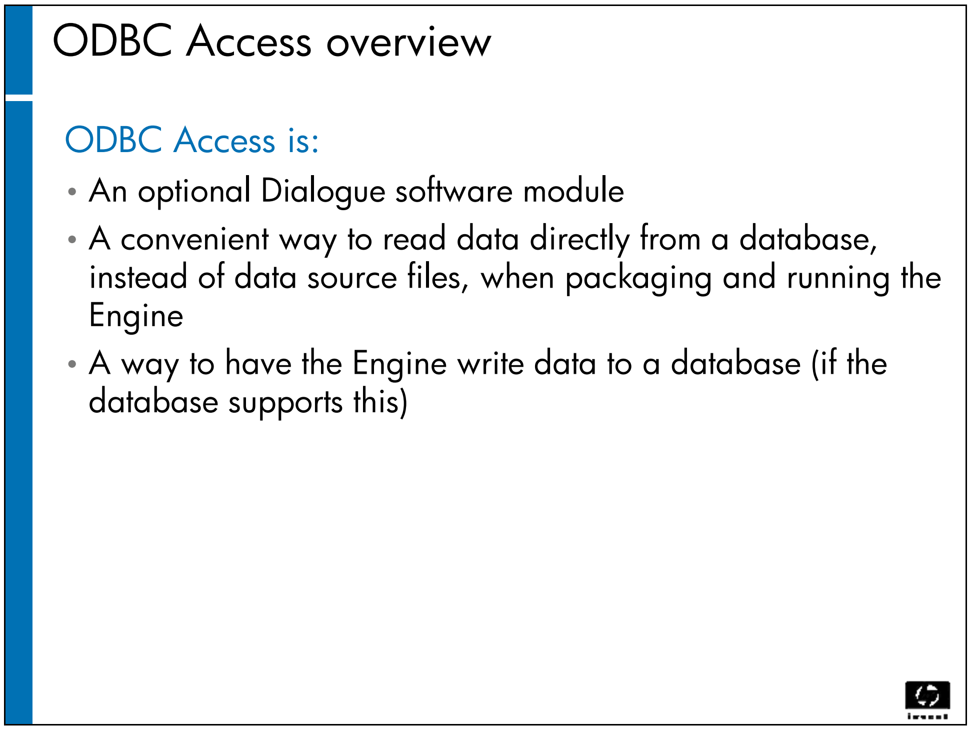


In this chapter

Objectives:

- Describe and discuss ODBC access and relational database basics
- Create and define a Reference file for ODBC Access operations.

9.2 ODBC Access Overview

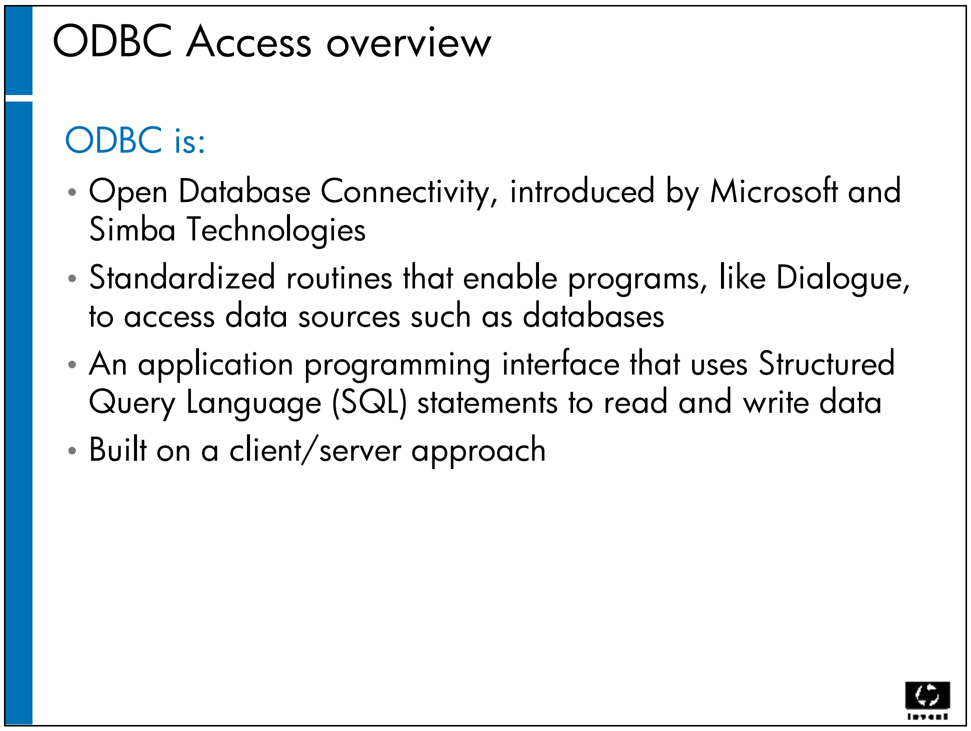


ODBC Access overview

ODBC Access is:

- An optional Dialogue software module
- A convenient way to read data directly from a database, instead of data source files, when packaging and running the Engine
- A way to have the Engine write data to a database (if the database supports this)

To verify that the **ODBC Access** module is enabled on your system, drag **System Settings** to the Property Panel. The ODBC Access module appears under the **Advanced data access** area on the **Key** tab. If its check box is selected, you have access to the features in this chapter.



ODBC Access overview

ODBC is:

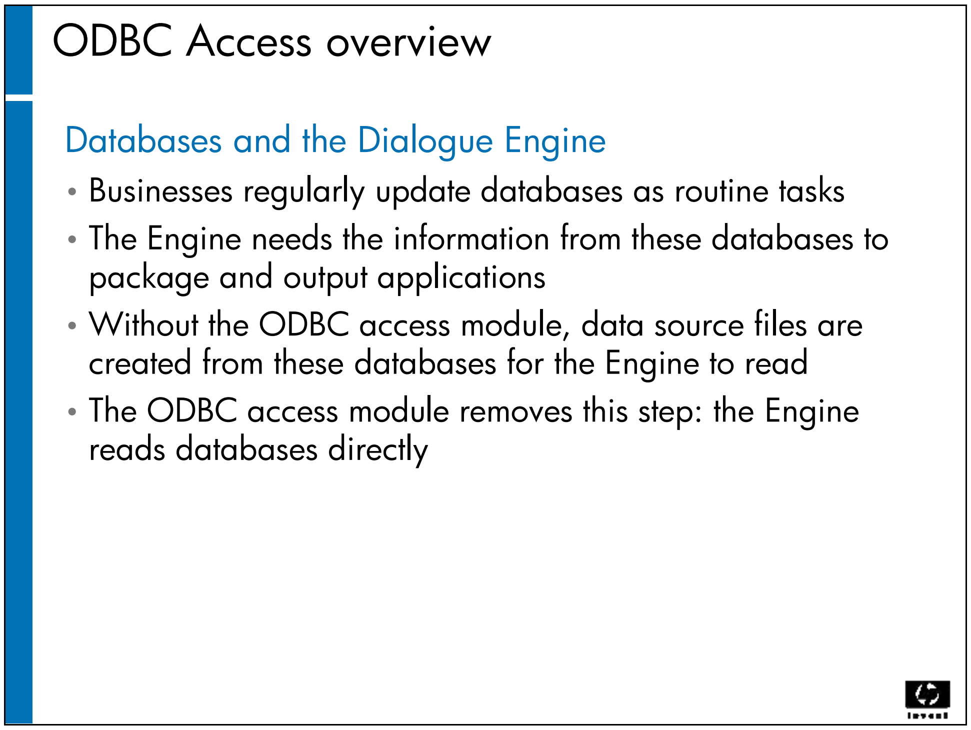
- Open Database Connectivity, introduced by Microsoft and Simba Technologies
- Standardized routines that enable programs, like Dialogue, to access data sources such as databases
- An application programming interface that uses Structured Query Language (SQL) statements to read and write data
- Built on a client/server approach

SQL is often pronounced "sequel" from its predecessor, Structured English Query Language (SEQUEL), designed at IBM in the mid 1970's. Although knowledge of SQL is helpful, you do *not* have to know SQL or ODBC to use the ODBC Access Module. Dialogue now provides support for SQL server 2005 (64 bit) as well earlier versions of SQL Server.

The advantage to a client/server architecture is that you can connect to multiple databases on your workstation – or *anywhere* in your organization. You can have Dialogue reach databases in another department, branch, headquarters, or foreign office if you have access permissions and network connections to them.

Databases and ODBC Access

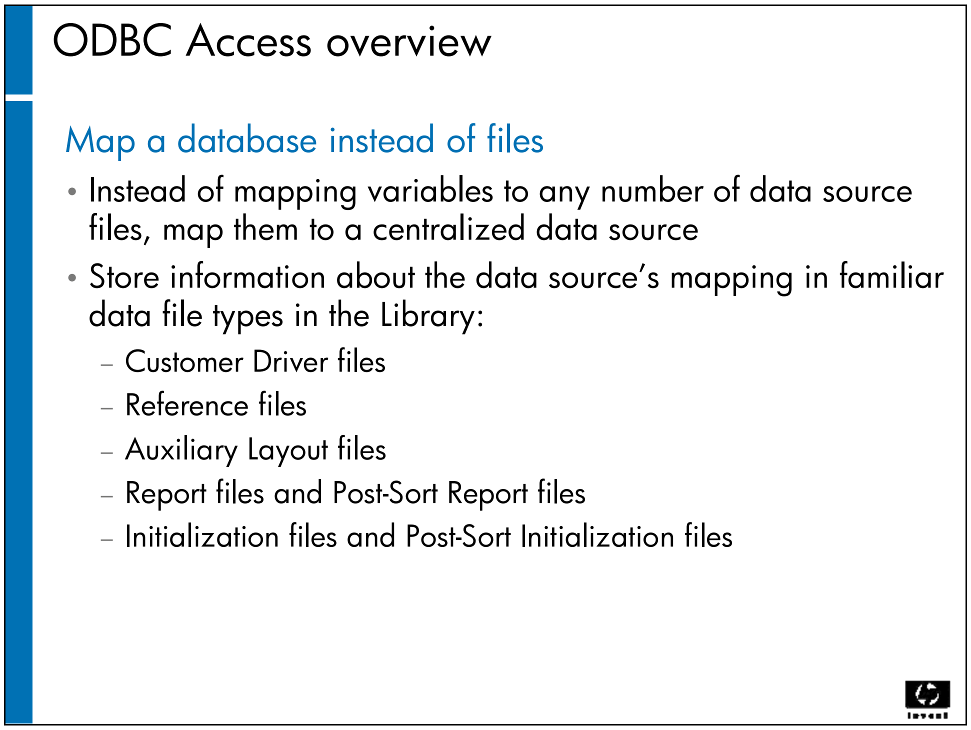
Most organizations operate multiple databases from multiple vendors, such as Oracle, Microsoft, or IBM – to name a few. They contain the latest customer information available.



ODBC Access overview

Databases and the Dialogue Engine

- Businesses regularly update databases as routine tasks
- The Engine needs the information from these databases to package and output applications
- Without the ODBC access module, data source files are created from these databases for the Engine to read
- The ODBC access module removes this step: the Engine reads databases directly



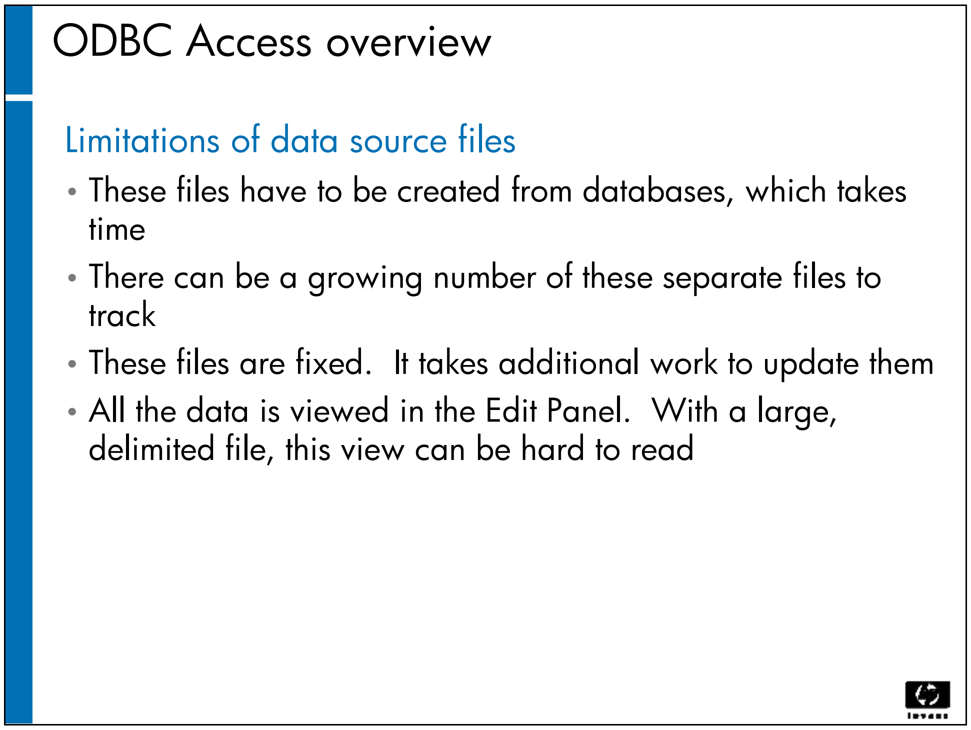
ODBC Access overview

Map a database instead of files

- Instead of mapping variables to any number of data source files, map them to a centralized data source
- Store information about the data source's mapping in familiar data file types in the Library:
  - Customer Driver files
  - Reference files
  - Auxiliary Layout files
  - Report files and Post-Sort Report files
  - Initialization files and Post-Sort Initialization files

Limitations of Data Source Files

Without the ODBC Access Module, you create data source files (in .txt, .dat, .pdf, or .xml format) from various databases. These flat files do a good job in providing data for Dialogue when you package and run the Engine. However, they do have a few limitations.

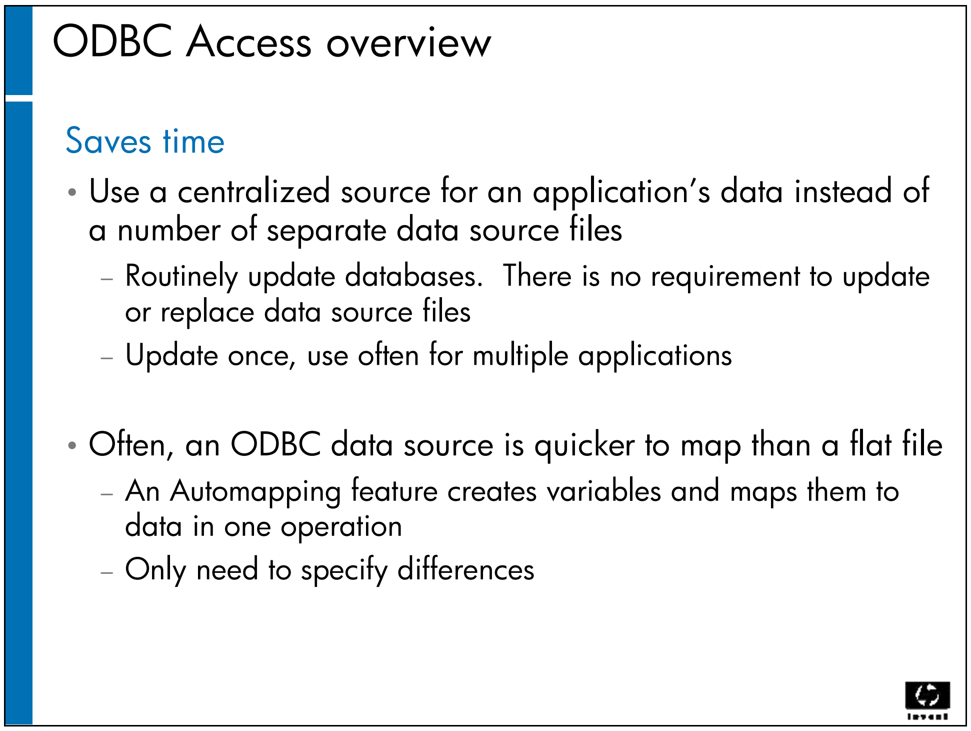


ODBC Access overview

Limitations of data source files

- These files have to be created from databases, which takes time
- There can be a growing number of these separate files to track
- These files are fixed. It takes additional work to update them
- All the data is viewed in the Edit Panel. With a large, delimited file, this view can be hard to read

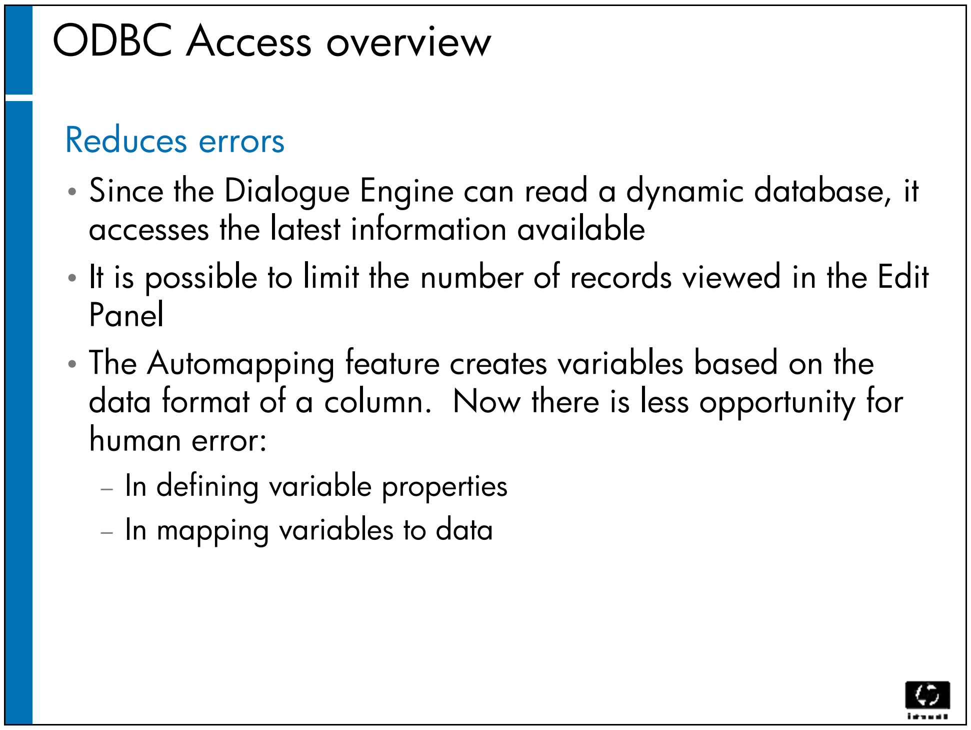
- You have to create these files from databases, which takes time. In large organizations, this may involve having to communicate with a different department than yours.
- There can be a growing number of these separate files to track. None of these data source files appear in the Library, so you have to devise ways of keeping track of these external files, which can get increasingly more time-consuming.
- These files are fixed. To keep them up to date, you have to replace them with new ones, update them with third-party software, or manually make changes. Then, after an update the information in the files is still static and could become outdated in a short period of time.
- You see all the data in the Edit Panel. Unless you set the **Limit Datamapping Records** text box (in the **Advanced** tab of data file properties) to a low number, you view multiple records as you map. With large delimited files, this view can be especially hard to read. With the ODBC Access module, you can view just one record, or any other number you choose.



ODBC Access overview

Saves time

- Use a centralized source for an application’s data instead of a number of separate data source files
  - Routinely update databases. There is no requirement to update or replace data source files
  - Update once, use often for multiple applications
- Often, an ODBC data source is quicker to map than a flat file
  - An Automapping feature creates variables and maps them to data in one operation
  - Only need to specify differences



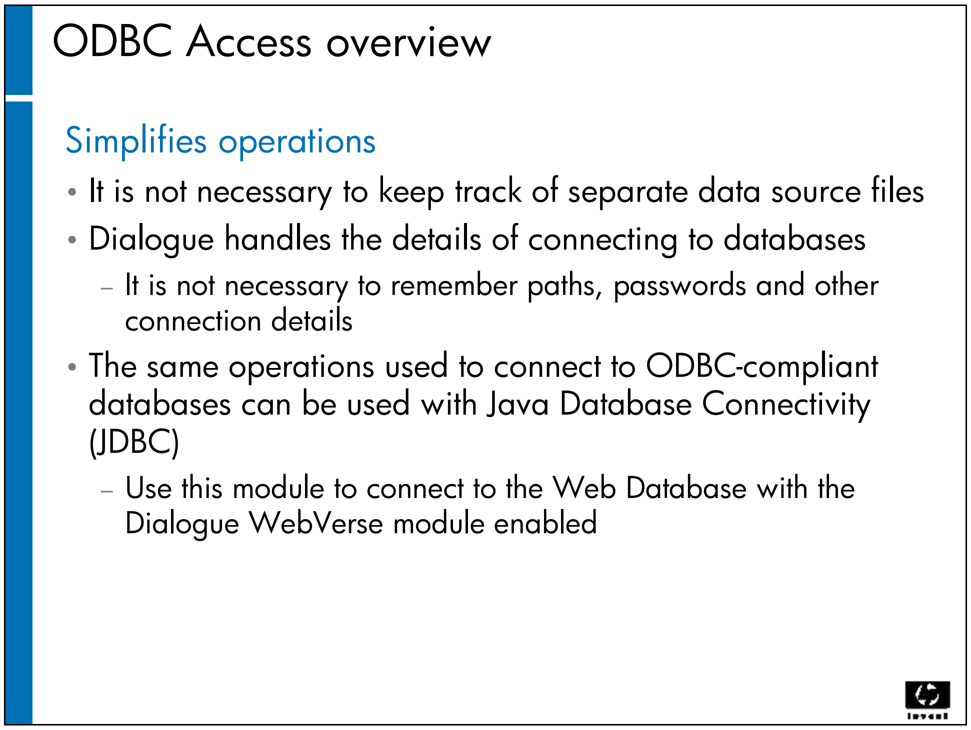
ODBC Access overview

Reduces errors

- Since the Dialogue Engine can read a dynamic database, it accesses the latest information available
- It is possible to limit the number of records viewed in the Edit Panel
- The Automapping feature creates variables based on the data format of a column. Now there is less opportunity for human error:
  - In defining variable properties
  - In mapping variables to data

With the ODBC Access module, you *can* continue to use data source files whenever you want. They are simply no longer a requirement.

For example, your organization might prefer having Report files continue to write to flat files rather than writing to a database. Dialogue offers the flexibility to use either destination.

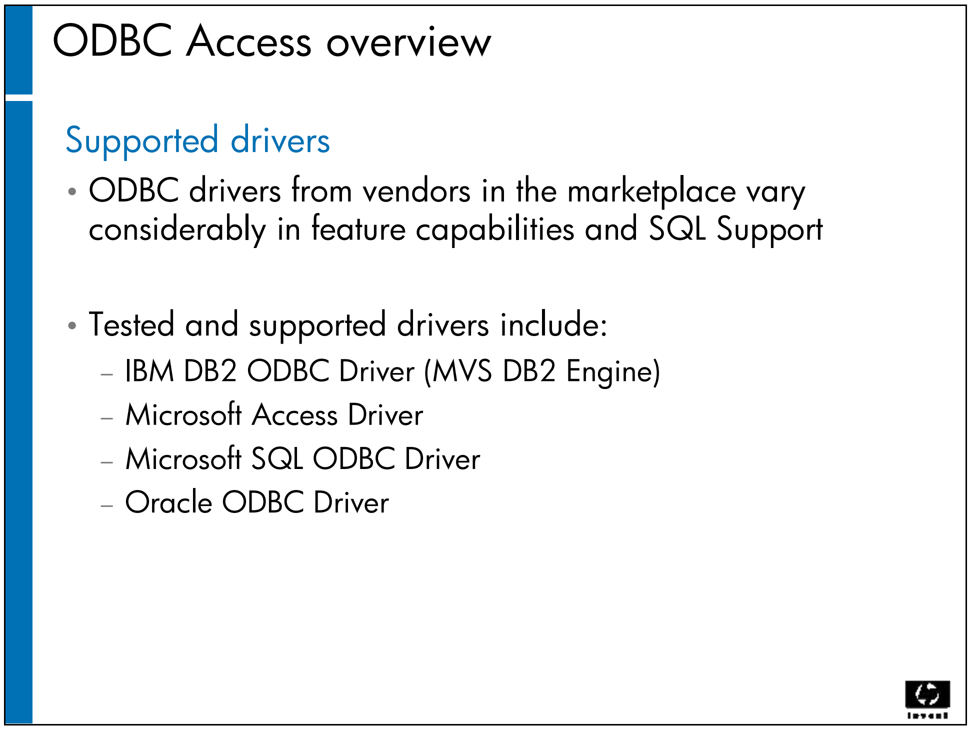


ODBC Access overview

Simplifies operations

- It is not necessary to keep track of separate data source files
- Dialogue handles the details of connecting to databases
  - It is not necessary to remember paths, passwords and other connection details
- The same operations used to connect to ODBC-compliant databases can be used with Java Database Connectivity (JDBC)
  - Use this module to connect to the Web Database with the Dialogue WebVerse module enabled

You can build on the functionality and operations in the **ODBC Access** module if you enable the **WebVerse** module.



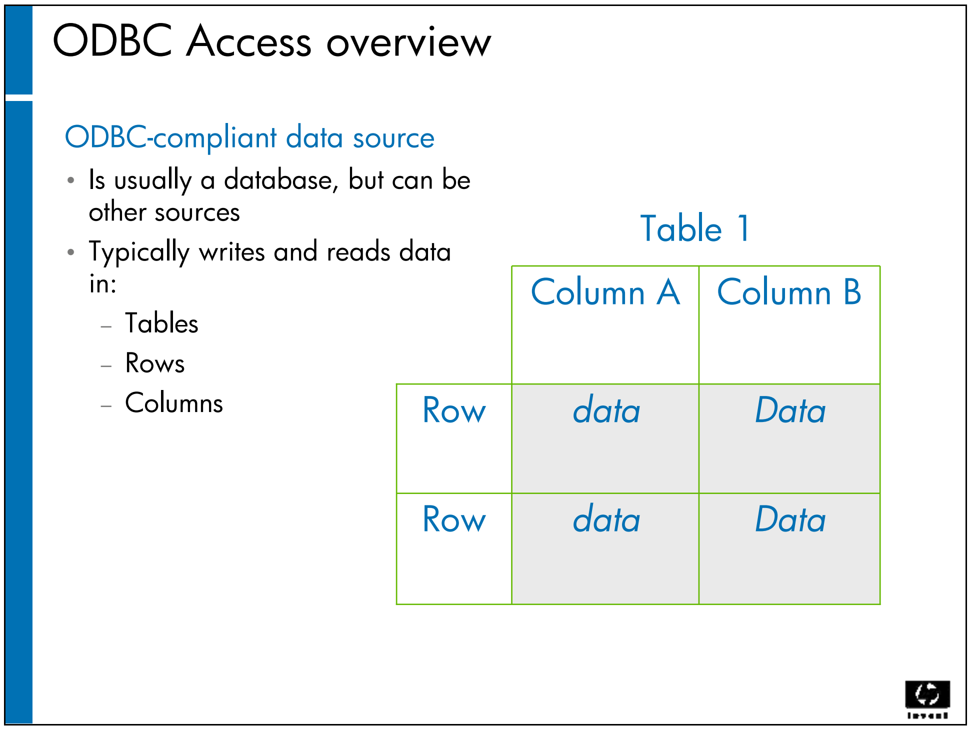
ODBC Access overview

Supported drivers

- ODBC drivers from vendors in the marketplace vary considerably in feature capabilities and SQL Support
- Tested and supported drivers include:
  - IBM DB2 ODBC Driver (MVS DB2 Engine)
  - Microsoft Access Driver
  - Microsoft SQL ODBC Driver
  - Oracle ODBC Driver

These drivers support connectivity and basic operations. Some drivers may not support all ODBC Access Module operations. For example, the Microsoft Access Driver does not support the **Update** function.

With the ODBC Access module, Dialogue can directly reach ODBC-compliant data sources, using a supported commercially available driver.



ODBC Access overview

ODBC-compliant data source

- Is usually a database, but can be other sources
- Typically writes and reads data in:
  - Tables
  - Rows
  - Columns

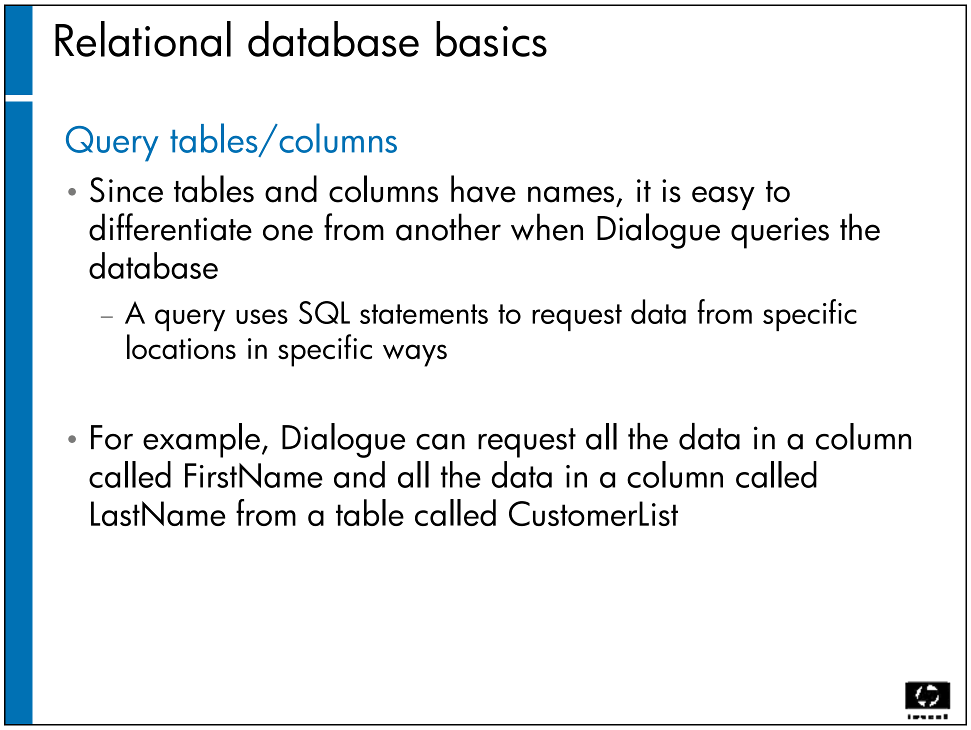
Table 1

	Column A	Column B
Row	*data*	*Data*
Row	*data*	*Data*

9.3 Relational Database Basics

To put in very simplistic terms:

- A **table** is a set of rows. Each table has a unique name.
- A **row** (also called a **record**) is a collection of data (supplied by multiple columns) that describe one specific entity. Rows do *not* have unique names.
- A **column** (also called a **field**) separates data in a row by different data types, such as integer, text string, or date. Each column has a unique name.

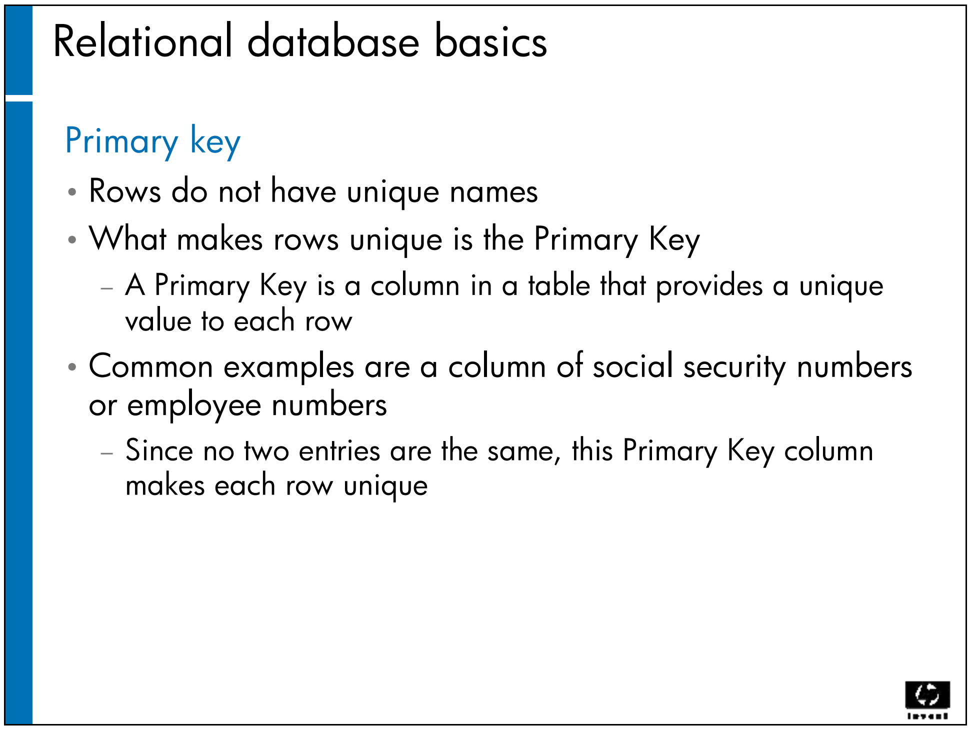


Relational database basics

Query tables/columns

- Since tables and columns have names, it is easy to differentiate one from another when Dialogue queries the database
  - A query uses SQL statements to request data from specific locations in specific ways
- For example, Dialogue can request all the data in a column called FirstName and all the data in a column called LastName from a table called CustomerList

However, rows (which typically grow more numerous than columns or tables) are not treated in the same way.



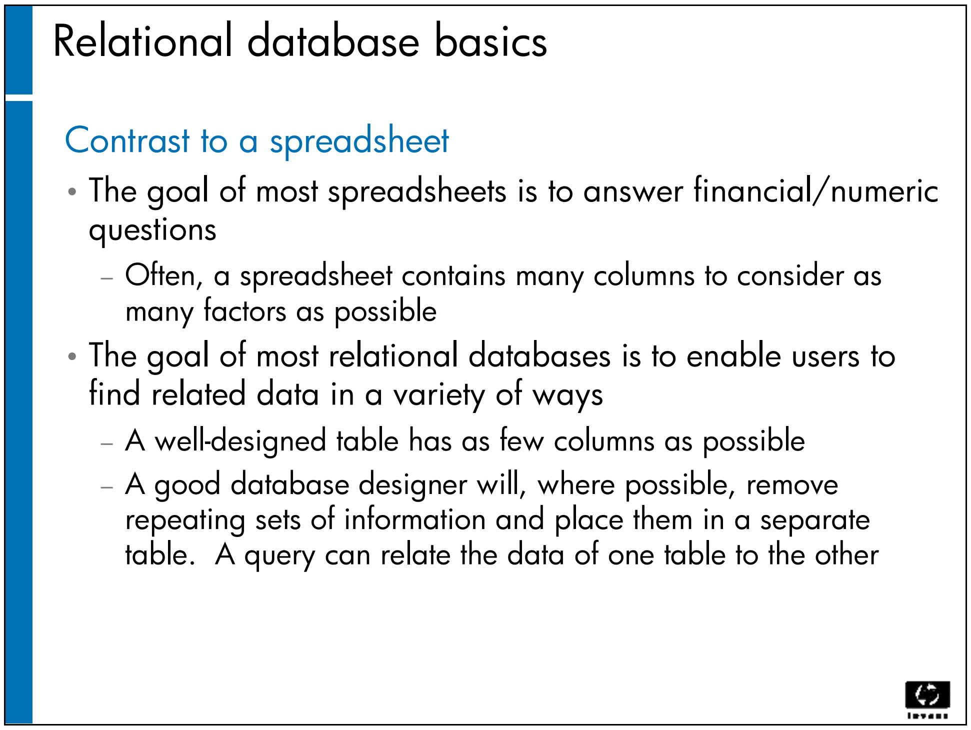
Relational database basics

Primary key

- Rows do not have unique names
- What makes rows unique is the Primary Key
  - A Primary Key is a column in a table that provides a unique value to each row
- Common examples are a column of social security numbers or employee numbers
  - Since no two entries are the same, this Primary Key column makes each row unique

Another common Primary Key is a column with autonumbering. With autonumbering, as you add a record to the table, the database automatically assigns a unique number to the Primary Key column. (Think of this as a dynamic version of the numbered far left column in a Microsoft Excel spreadsheet.) Often, the Primary Key is the leftmost column, but it can occur anywhere in a table.

With a Primary Key, there are additional ways for you to define a query. For example, you can set up a query to read all the data in the Sales and NumberOfOrders columns in the CustomerList Table for customers having a **CustomerNumber** between 0000 and 0012. This example assumes that each separate customer has a unique value. In this example, the **CustomerNumber** column is very likely the Primary Key column.

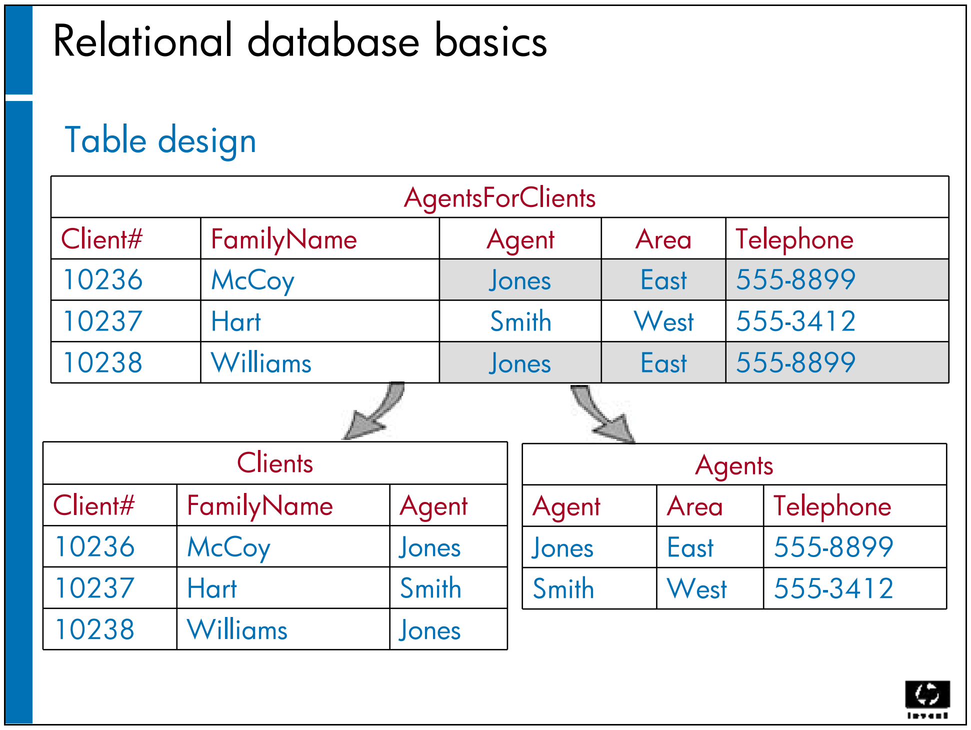


Relational database basics

Contrast to a spreadsheet

- The goal of most spreadsheets is to answer financial/numeric questions
  - Often, a spreadsheet contains many columns to consider as many factors as possible
- The goal of most relational databases is to enable users to find related data in a variety of ways
  - A well-designed table has as few columns as possible
  - A good database designer will, where possible, remove repeating sets of information and place them in a separate table. A query can relate the data of one table to the other

The top table in the following slide is inefficient. The shaded areas are redundant. One problem with redundant data is updating: if *Jones* leaves the company or takes charge of another service area or has to change phone numbers, you must to go to multiple places to make the change. In a large database, finding all occurrences could be labor-intensive.



Relational database basics

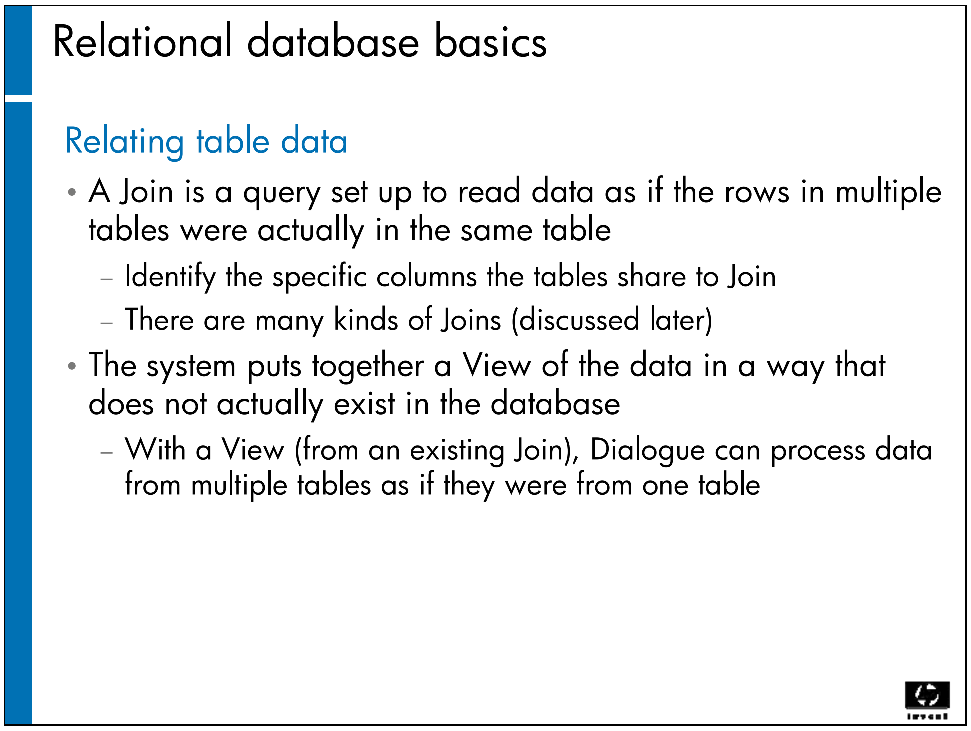
Table design

AgentsForClients				
Client#	FamilyName	Agent	Area	Telephone
10236	McCoy	Jones	East	555-8899
10237	Hart	Smith	West	555-3412
10238	Williams	Jones	East	555-8899

Clients		
Client#	FamilyName	Agent
10236	McCoy	Jones
10237	Hart	Smith
10238	Williams	Jones

Agents		
Agent	Area	Telephone
Jones	East	555-8899
Smith	West	555-3412

The two bottom tables are more efficient. Now, any change to any client or agent only involves *one row*. This design also speeds queries by Dialogue or any other program needing to read this information in the database. Note that both the Clients table and the Agents table have an Agent column. This common column serves as a way for both tables to *relate* to each other.



Relational database basics

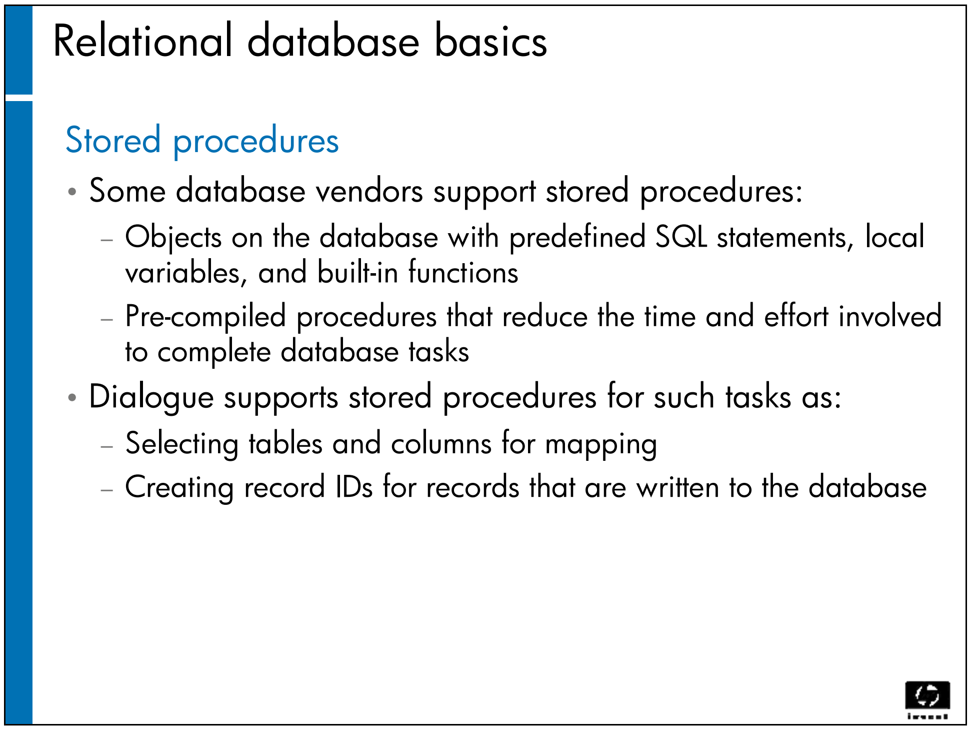
Relating table data

- A Join is a query set up to read data as if the rows in multiple tables were actually in the same table
  - Identify the specific columns the tables share to Join
  - There are many kinds of Joins (discussed later)
- The system puts together a View of the data in a way that does not actually exist in the database
  - With a View (from an existing Join), Dialogue can process data from multiple tables as if they were from one table

In the example on the previous page, both bottom tables share the same column, *Agents*. By *joining* the two tables at the Agent column, Dialogue can query all the agent/client information in this sample database in a single *view*.

Stored Procedure

Stored Procedures, often referred to as *stored procs*, are pre-designed procedures. These optional timesavers save keystrokes and can help to reduce human error. Stored procedures are not available with all databases. However, you can use DB2 stored procedures to do complex SQL coding on ODBC Access data files.



Relational database basics

Stored procedures

- Some database vendors support stored procedures:
  - Objects on the database with predefined SQL statements, local variables, and built-in functions
  - Pre-compiled procedures that reduce the time and effort involved to complete database tasks
- Dialogue supports stored procedures for such tasks as:
  - Selecting tables and columns for mapping
  - Creating record IDs for records that are written to the database

Keep Queries Precise

One goal of ODBC operation is to try to makes queries to the database as precise as possible. On top of normal database latency, you do not want to slow down reads and writes by having the database send information Dialogue does not need. This becomes more important if the database is in a remote location with slow-speed (or shared) communication paths.

One way to make ODBC operations more efficient is to set up **joins**. A single query reads columns in multiple tables more efficiently than multiple queries.

Another help is to limit the data the database will send back to Dialogue through **filters**. With filtering you specify conditions to separate out unneeded data. The query then skips over data the Engine does not need to process your application. A typical filter compares specific column *data* with a variable's *value* using some *condition* (such as *is not equal to*, or *is less than*).

9.4 Exercise: Create ODBC Data Files

The *ODBC Access Module* provides an additional data **File Format**. The **ODBC Data Source** option is available when you create a data file in the Library. This data file *format* can be used with many **File Types**:

- Customer driver file
- Reference file
- Auxiliary layout file
- Initialization file
- Post-Sort initialization file
- Report file
- Post-Sort report file

Instead of mapping variables to a data source file, you map variables to database tables and columns with these files.

A. Create a Reference File

In this section you will create a Reference file you will map in a later exercise.

1. Right-click the **Data Files** heading under the **Exercise 09-10 ODBC Access** folder.
2. Choose **New Data File** from the shortcut menu.

The **New Data File** dialog box appears.

3. Type `ODBC Reference File` in the **Name** text box.

NOTE: Ensure no filters are set in the variable palette.

4. In the **File type** drop-down list, choose **Reference file**.
5. In the **File format** drop-down list, choose **ODBC data source**.
6. Click **Finish**.
   The file appears in the Property Panel for you to define.

Basic Tab

With a **File type** of a Reference File, additional properties appear inside the **File type** area on the **Basic** tab. Before you define the properties in the **File type** area, you must decide if *one* variable defines the key to the Reference File or if *multiple* variables in a *Auxiliary Layout File* define the key. In this exercise you will use a single variable as the key.

7. Click the **Basic** tab.
8. From the **Reference key layout** drop-down list, choose **Single variable**.
9. At **Key**, click the button and select **AccountNumber**, from the **Exercise 09-10 ODBC Access** folder.
10. From the **Access** drop-down list, choose **Memory**.
    Your Access choice determines *how* the Engine will store and read data mapped in the Reference file. The **Memory** option offers the fastest processing, but uses the most resources.
11. Click the **Advanced** tab.

Advanced Tab

On the **Advanced** tab, the **IO Time** property becomes available if the **File Type** is a *Reference File* or *Report File*. Depending on your selection in the drop-down list, the **Section Name** text box may also be available.

In the **IO Time** drop-down list you specify:

- For a *Reference* data file, the time when the Engine will **read** data. You choose *how* the Engine reads information in the **Access** drop-down list in the **Basic** tab.
- For a *Report* data file, the time when the Engine will **write** data.

The timing options vary by the data file type.

12. From the **IO Time** drop-down list choose **After initial customer data**.
    The Reference File will provide account totals after customer data has been processed.

Data Source Tabs

13. Click the Test Data Source tab.
14. Click the button, select **ODBC Source** from the list, and click **OK**.
    The DSN name appears in the text box.
15. Click the **Production Data Source** tab.
16. Click the button, select **ODBC Source** from the list, and click **OK**.
    The DSN name appears in the text box.
17. Press **Test Connection** and click **OK** to remove the message.
18. Save and close the Property Panel

Chapter 12: Dynamic Content Import

12.1 In this Chapter

In this chapter, you will learn about **Dynamic Content Import**. You will insert a signature image onto the letter page using a **Placeholder Variable**.

Objectives

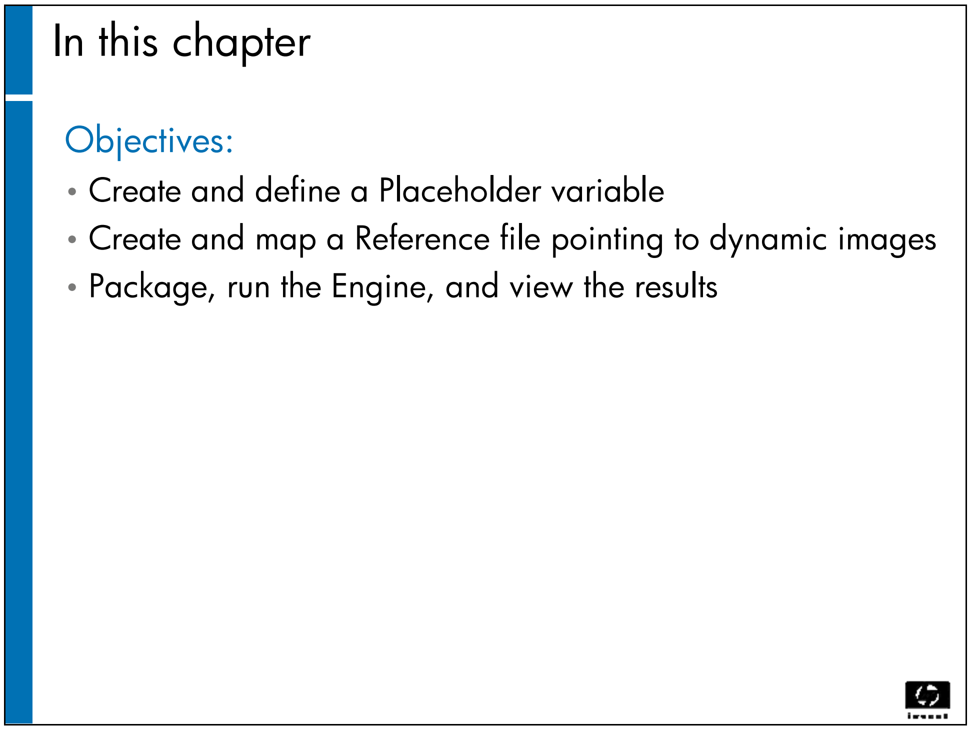
By the end of this chapter you will:

- Create and define a Placeholder Variable.
- Create and map a Reference File pointing to dynamic images.
- Package, run the Engine, and view the results.

For more information

For more information, refer to the following Dialogue documentation.

- Dynamically Importing Content guide Dynamic.pdf
- System Administration guide   System.pdf

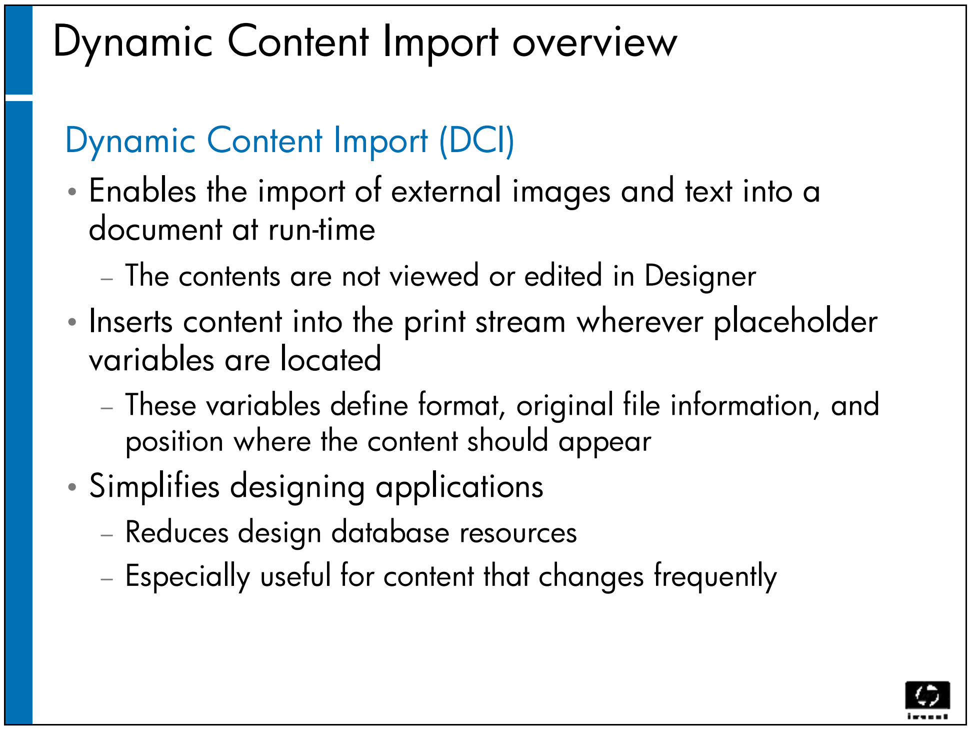


In this chapter

Objectives:

- Create and define a Placeholder variable
- Create and map a Reference file pointing to dynamic images
- Package, run the Engine, and view the results

12.2 Dynamic Content Import Overview

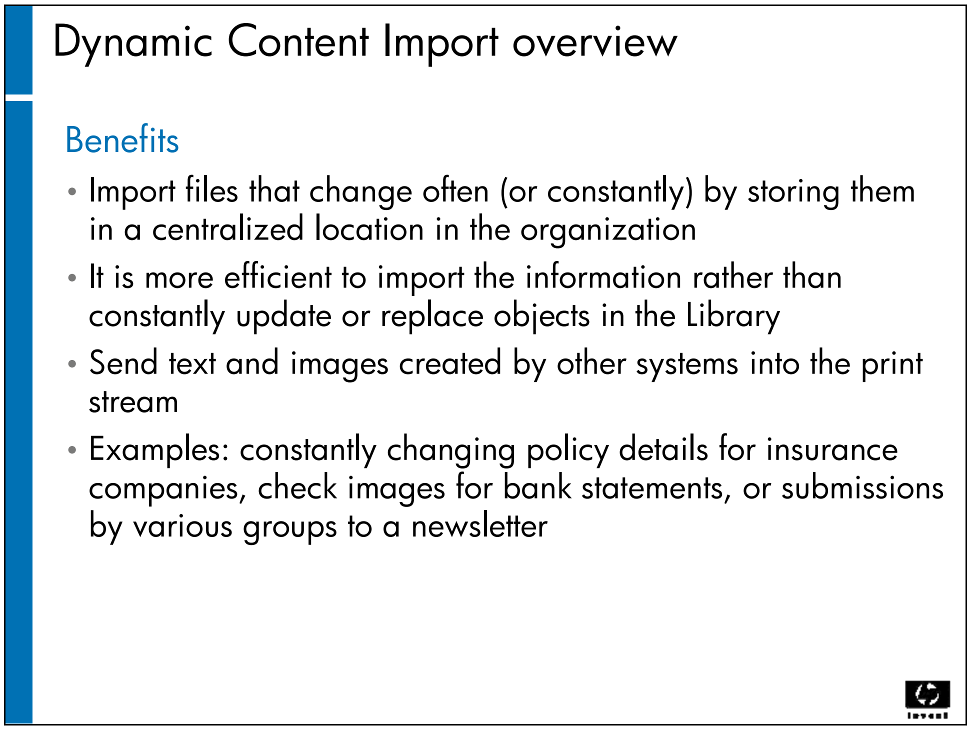


Dynamic Content Import overview

Dynamic Content Import (DCI)

- Enables the import of external images and text into a document at run-time
  - The contents are not viewed or edited in Designer
- Inserts content into the print stream wherever placeholder variables are located
  - These variables define format, original file information, and position where the content should appear
- Simplifies designing applications
  - Reduces design database resources
  - Especially useful for content that changes frequently

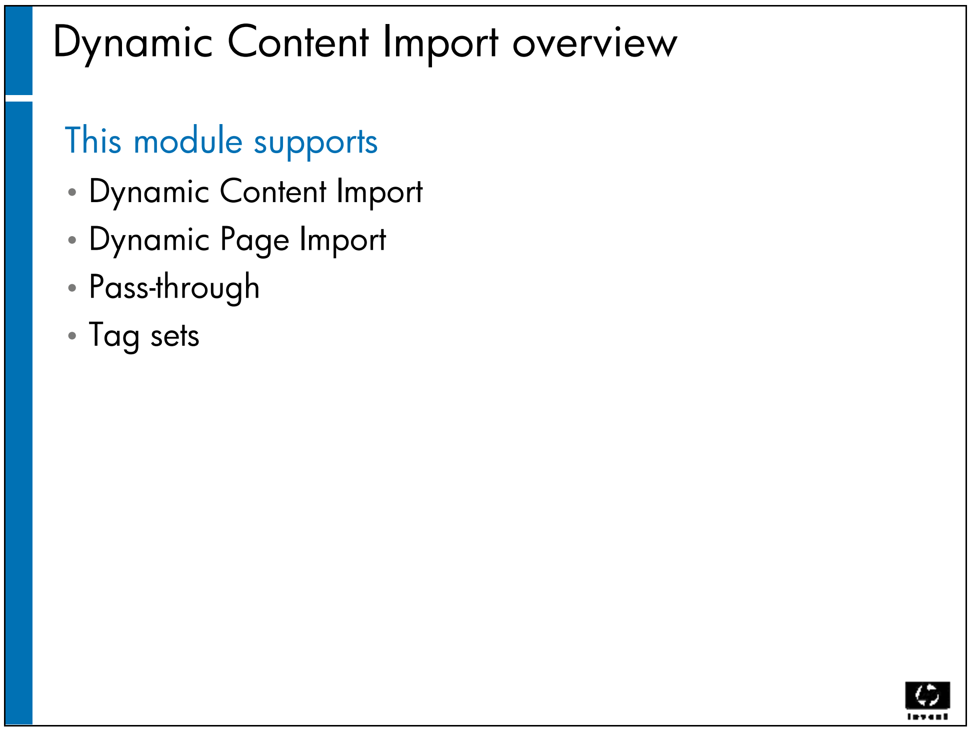
To verify that the **Dynamic Content Import** module is enabled on your system, drag **System Settings** to the Property Panel. This module appears under the **Document Creation** area on the **Key** tab. If its check box is selected, you have access to the features in this chapter.



Dynamic Content Import overview

Benefits

- Import files that change often (or constantly) by storing them in a centralized location in the organization
- It is more efficient to import the information rather than constantly update or replace objects in the Library
- Send text and images created by other systems into the print stream
- Examples: constantly changing policy details for insurance companies, check images for bank statements, or submissions by various groups to a newsletter



Dynamic Content Import overview

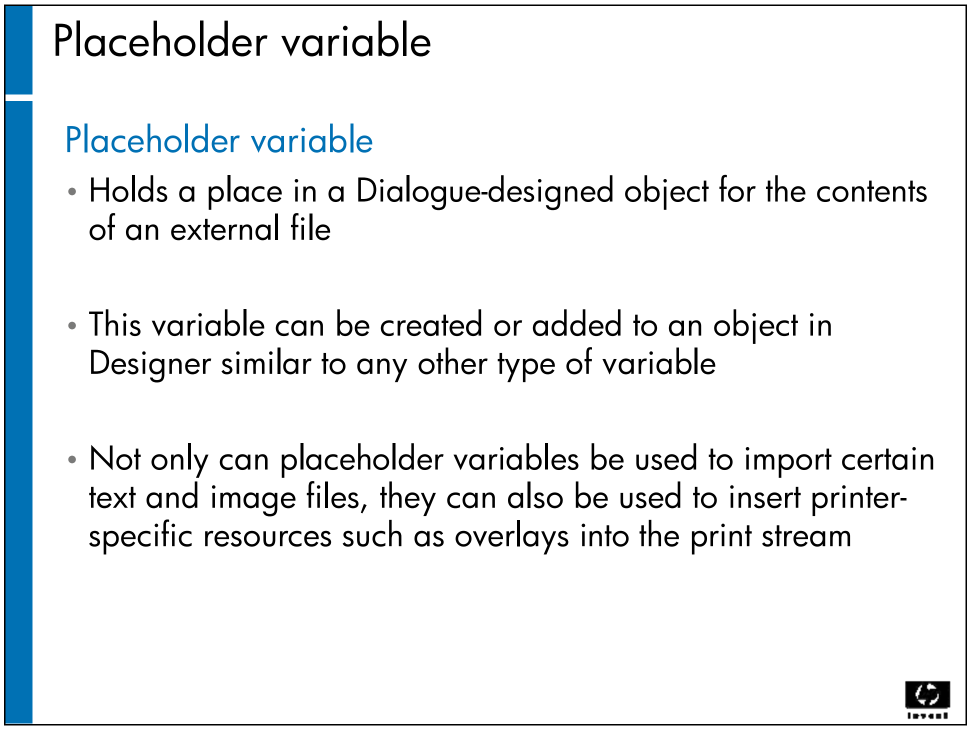
This module supports

- Dynamic Content Import
- Dynamic Page Import
- Pass-through
- Tag sets

You will explore Dynamic Content Import and Dynamic Page Import in this chapter. Details on Tag Sets appear in the following chapter. You can use the new PDF form pre-fill module to dynamically import and populate PDF/XFA (XML Forms Architecture) forms. XFA is a set of specifications for building interactive forms used by Adobe Acrobat 6.0 and newer. Forms must contain XFA to be populated.

12.3 Placeholder Variable

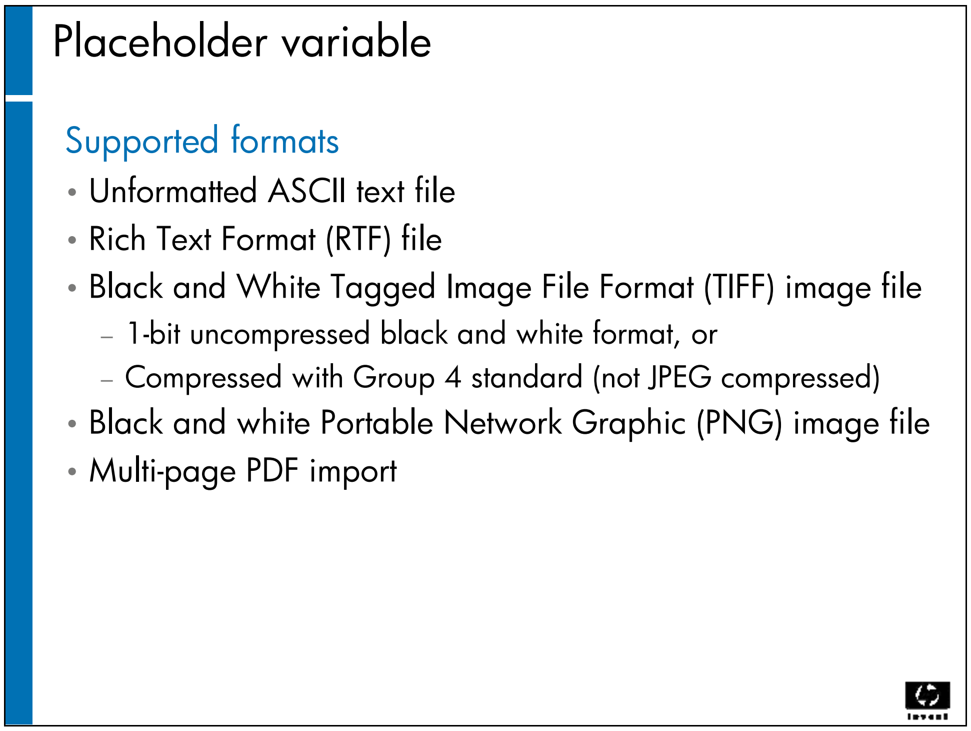
The main tool you use in Dynamic Content Import operations is a *Placeholder Variable*.



Placeholder variable

Placeholder variable

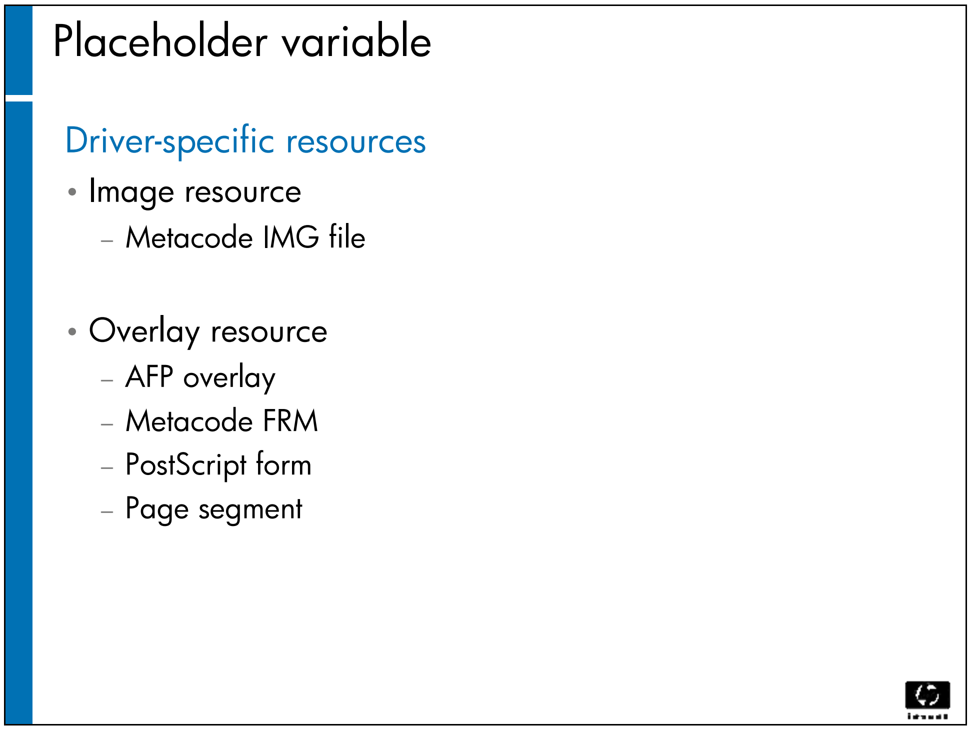
- Holds a place in a Dialogue-designed object for the contents of an external file
- This variable can be created or added to an object in Designer similar to any other type of variable
- Not only can placeholder variables be used to import certain text and image files, they can also be used to insert printer-specific resources such as overlays into the print stream



Placeholder variable

Supported formats

- Unformatted ASCII text file
- Rich Text Format (RTF) file
- Black and White Tagged Image File Format (TIFF) image file
  - 1-bit uncompressed black and white format, or
  - Compressed with Group 4 standard (not JPEG compressed)
- Black and white Portable Network Graphic (PNG) image file
- Multi-page PDF import



Placeholder variable

Driver-specific resources

- Image resource
  - Metacode IMG file

- Overlay resource
  - AFP overlay
  - Metacode FRM
  - PostScript form
  - Page segment

Using Placeholder Variables

You follow this sequence of steps to use Dynamic Content Import:

- Create a placeholder variable in the Library. Specify in its properties whether this variable will import or pass-through the content.
- Place this placeholder variable on a message or page.
- Create a data source file that identifies the path and name of the external file.
- Map the placeholder variable to the data source file using a Dialogue data file, such as an Initialization File.
- Package and run the Engine.

Additional information on **Dynamic Content Import** appears in the *302 Creating Long Documents* Dialogue training course.

12.3 Exercise: Dynamic Content Import

In this exercise you will create a Placeholder variable to import TIFF images of signatures for the letter application you have been using in this course.

A. Create and Define a Placeholder Variable

1. In the Library, expand the Exercise 12-13 Dynamic Content Import folder.
2. Under this folder right-click the Data Dictionary heading.
3. Choose New Variable in the shortcut menu.
   The New Variable dialog box appears.
4. Enter `SignaturePlaceholder` in the Name text box.
5. In the Type drop-down list choose Placeholder.

When you select this **Type**, another drop-down list appears.

6. In the adjacent drop-down list choose **B&W TIFF (G4 or uncomp)** as the type of file the placeholder will import.
7. Click **Finish**.

The variable appears in the Property Panel. You define a Placeholder Variable as you do any other variable.

8. On the **Basic** tab, choose **File only** from the **Source** drop-down list.
9. Choose **Before each customer** as the **Reset Time**.
10. Click the **Placeholder** tab.
11. Clear the check box entitled **The file for each customer is unique**.
12. Save and close the Property Panel.

B. Create and Define a Reference File

1. In the Library, under the **Exercise 12-13 Dynamic Content Import** folder, right-click the **Data Files** heading.
2. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
3. Enter `Placeholder Use` in the **Name** field.
4. Choose **Reference file** as the **File type**.
5. Choose **Delimited data file** as the **File format**.
6. Click **Finish**.
   The data file appears in the Property Panel for you to define.

7. On the **Basic** tab, select **CustomerState** from the **Exercise 12-13 Dynamic Content Import** folder as the **Key** variable.
8. Ensure the **Access** drop-down list displays the default of **Memory**.
9. In the **File Format** area, enter a comma as the **Delimiter**.
10. Ensure that the **How to identify customers in the data file** area displays the default of **Each customer has the same number of records**. Ensure that the **Number of records per customer** property has the default of **1**.
11. Click the **Test Data Source** tab.
12. In the **File to use in test mode** field, enter (or browse to):
    `C:\210 Advanced Data Concepts\Data Files\Area Managers Sig.dat`
13. Click the **Production Data Source** tab.
14. In the **File to use in Production** property, enter: `EXsig`
15. Save and close the Property Panel.

C. Map the Reference File

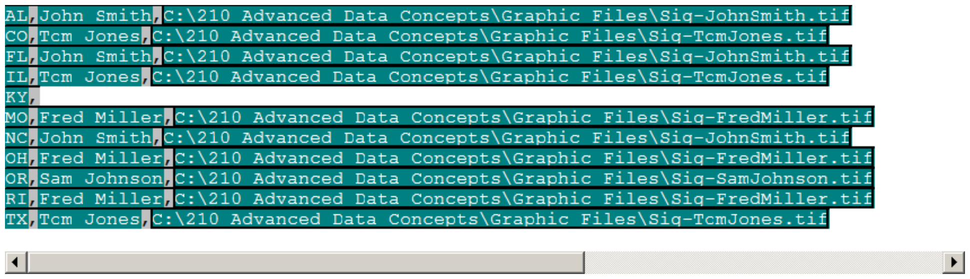
1. Expand the Data Files heading under the Exercise 12-13 Dynamic Content Import folder.
2. Drag the Placeholder Use data file to the Edit Panel.
3. Map the CustomerState variable to the first data area. No changes are needed in the Data Area Properties dialog box.

NOTE: This is the *Key* variable for this Reference File.

4. Map the **DistrictManager** variable to the second data area.
5. Map the **SignaturePlaceholder** variable you created earlier in this exercise to the third data area.

NOTE: Notice that the third data area specifies the location of a TIFF file. Mapping the placeholder variable to this data area tells Dialogue to insert the file at the location of the variable on the Design page.

Mapped file in the Edit Panel



```
AL,John Smith,C:\210 Advanced Data Concepts\Graphic Files\Sig-JohnSmith.tif
CO,Tom Jones,C:\210 Advanced Data Concepts\Graphic Files\Sig-TomJones.tif
FL,John Smith,C:\210 Advanced Data Concepts\Graphic Files\Sig-JohnSmith.tif
IL,Tom Jones,C:\210 Advanced Data Concepts\Graphic Files\Sig-TomJones.tif
KY,
MO,Fred Miller,C:\210 Advanced Data Concepts\Graphic Files\Sig-FredMiller.tif
NC,John Smith,C:\210 Advanced Data Concepts\Graphic Files\Sig-JohnSmith.tif
OH,Fred Miller,C:\210 Advanced Data Concepts\Graphic Files\Sig-FredMiller.tif
OR,Sam Johnson,C:\210 Advanced Data Concepts\Graphic Files\Sig-SamJohnson.tif
RI,Fred Miller,C:\210 Advanced Data Concepts\Graphic Files\Sig-FredMiller.tif
TX,Tom Jones,C:\210 Advanced Data Concepts\Graphic Files\Sig-TomJones.tif
```

6. Save and close the Edit Panel.

D. Add the Variable to the Design Page

1. Expand the Pages heading under the Exercise 12-13 Dynamic Content Import folder.
2. Drag the Placeholder Page to the Edit Panel to open it in Designer.
3. In Designer, click the letter body text box to go into edit mode.
4. Place your cursor in the line beneath Sincerely in the closing.
5. Press ENTER.
6. Open the Variable Panel if it is not already open.
7. Highlight the SignaturePlaceholder variable you created and double-click.
8. Save your changes and close Designer.

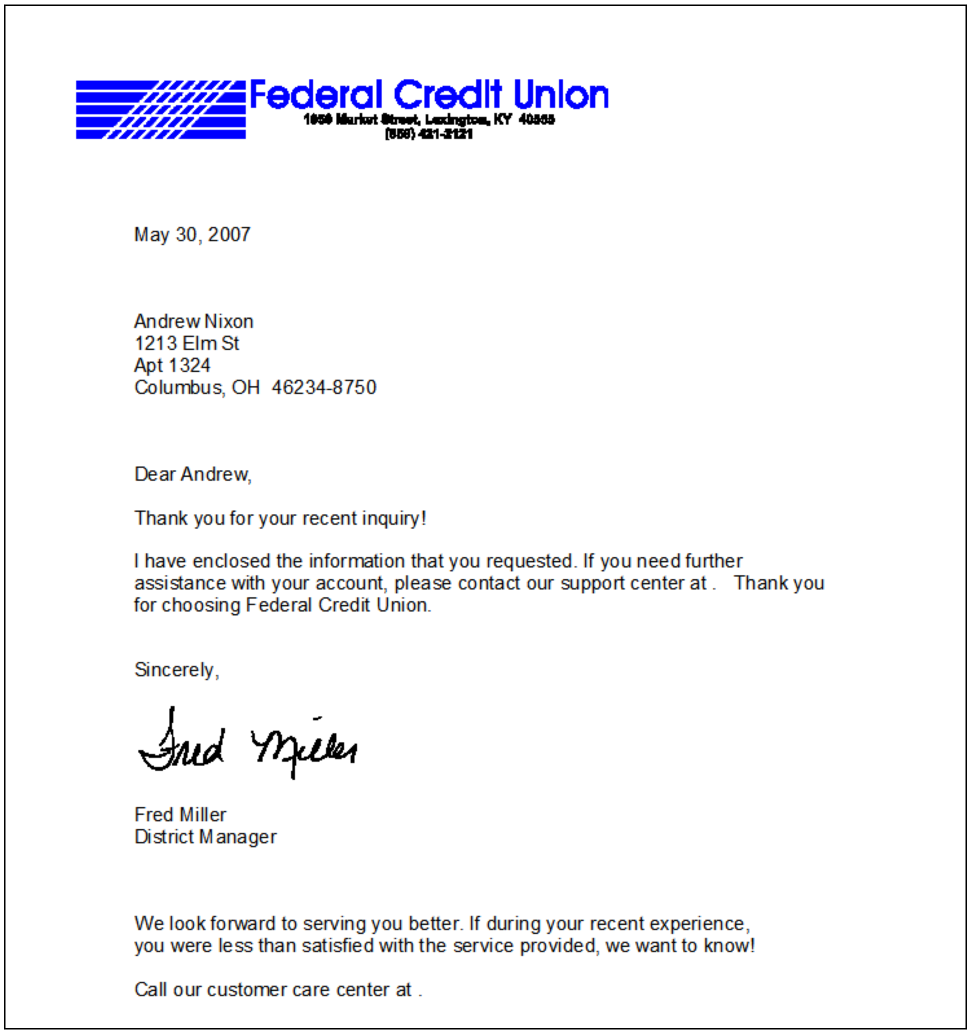
E. Add Objects to an Application

1. Expand the **Applications** heading under the **Exercise 12-13 Dynamic Content Import** folder so you can view the Placeholder Application.
2. Drag the **Placeholder Use** data file from the **Data Files** heading to the **Placeholder Application**. Place the data files in the following order; Initialization File, Customer Driver File, and Placeholder Use File.

F. Package the Application

1. Package and run the application.
2. Review the results in the Exstream Viewer.

Composed Page



Federal Credit Union
1050 Market Street, Lexington, KY 40555
(555) 421-2121

May 30, 2007

Andrew Nixon
1213 Elm St
Apt 1324
Columbus, OH 46234-8750

Dear Andrew,

Thank you for your recent inquiry!

I have enclosed the information that you requested. If you need further assistance with your account, please contact our support center at .   Thank you for choosing Federal Credit Union.

Sincerely,

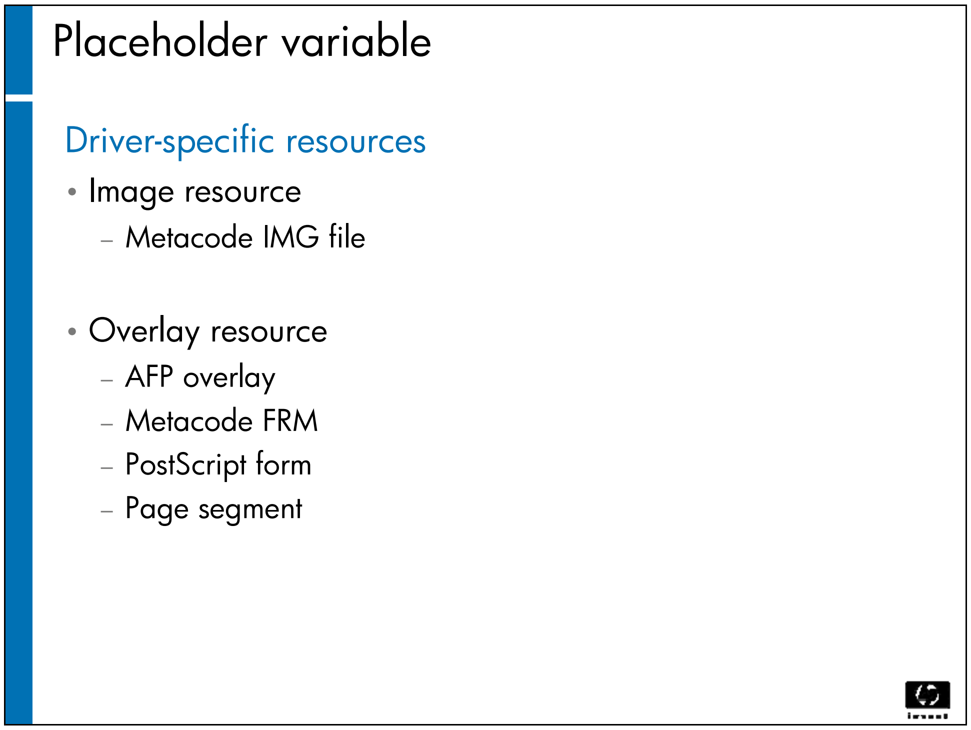
Fred Miller

Fred Miller
District Manager

We look forward to serving you better. If during your recent experience, you were less than satisfied with the service provided, we want to know!

Call our customer care center at .

12.5 Resource Management Features



Placeholder variable

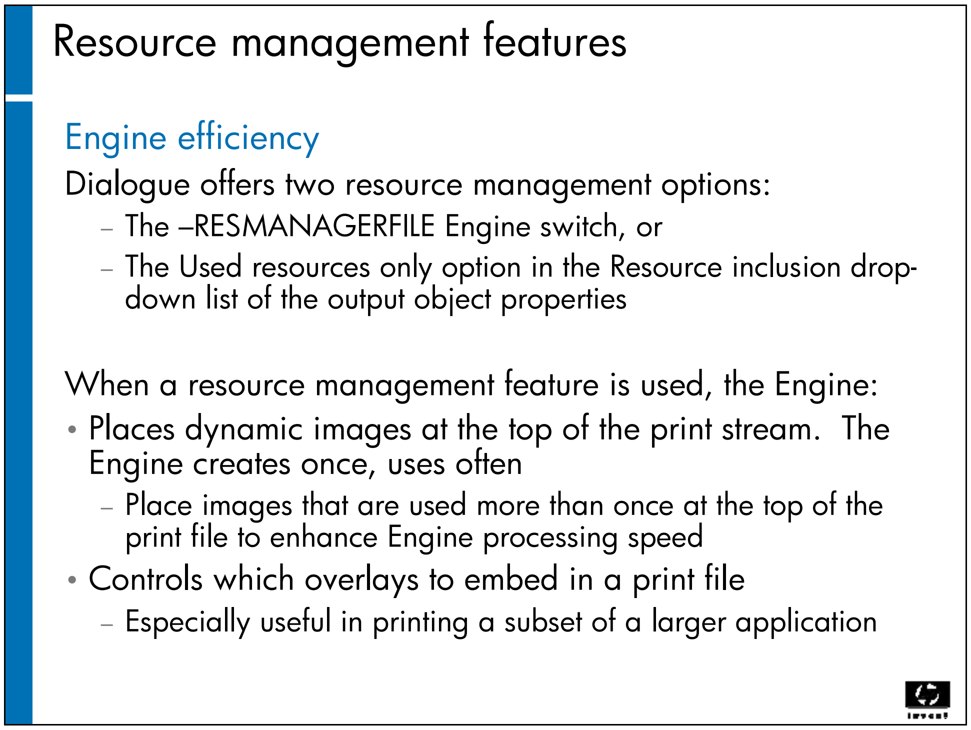
Driver-specific resources

- Image resource
  - Metacode IMG file
- Overlay resource
  - AFP overlay
  - Metacode FRM
  - PostScript form
  - Page segment

An inaccurate **image** list may result in the same image being placed on every page.

An inaccurate **overlay** list can result in an unusable print stream (with PostScript, VPS, VIPP, PPML, and PDF drivers) or can have missing static page content (AFP).

Specifications



Resource management features

Engine efficiency

Dialogue offers two resource management options:

- The –RESMANAGERFILE Engine switch, or
- The Used resources only option in the Resource inclusion drop-down list of the output object properties

When a resource management feature is used, the Engine:

- Places dynamic images at the top of the print stream. The Engine creates once, uses often
  - Place images that are used more than once at the top of the print file to enhance Engine processing speed
- Controls which overlays to embed in a print file
  - Especially useful in printing a subset of a larger application

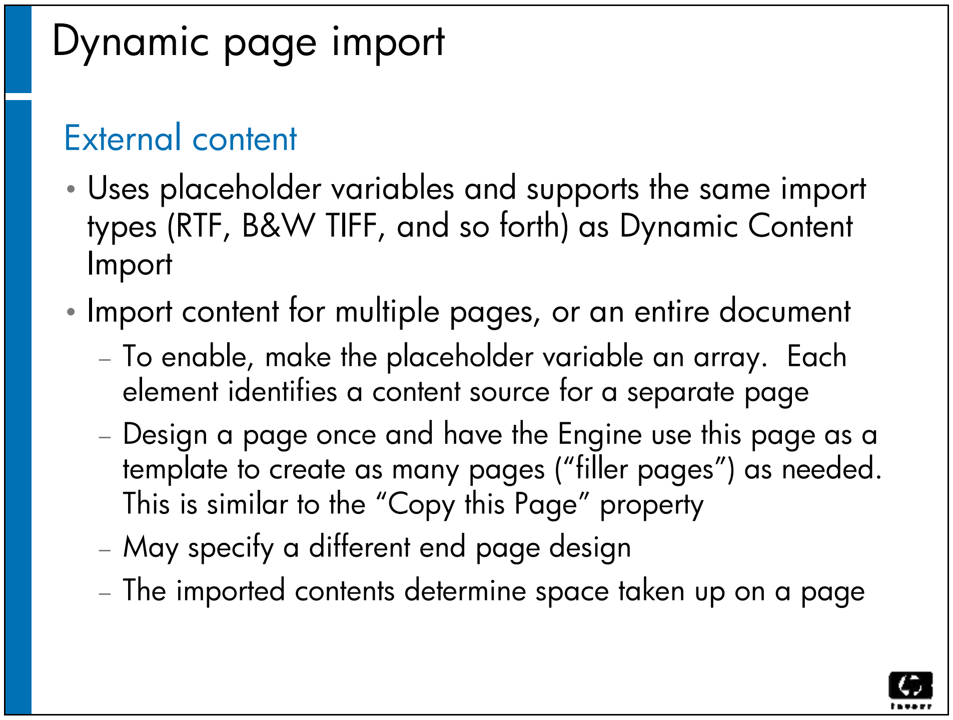
The Engine creates four lists:

- Images
- Page overlays
- Message overlays
- Template overlays

The Engine builds separate lists for these three overlay types. If no list is specified for a particular overlay all overlays of that type are embedded in the file. If a list is specified, only those overlays whose names are on the list are embedded.

12.6 Dynamic Page Import

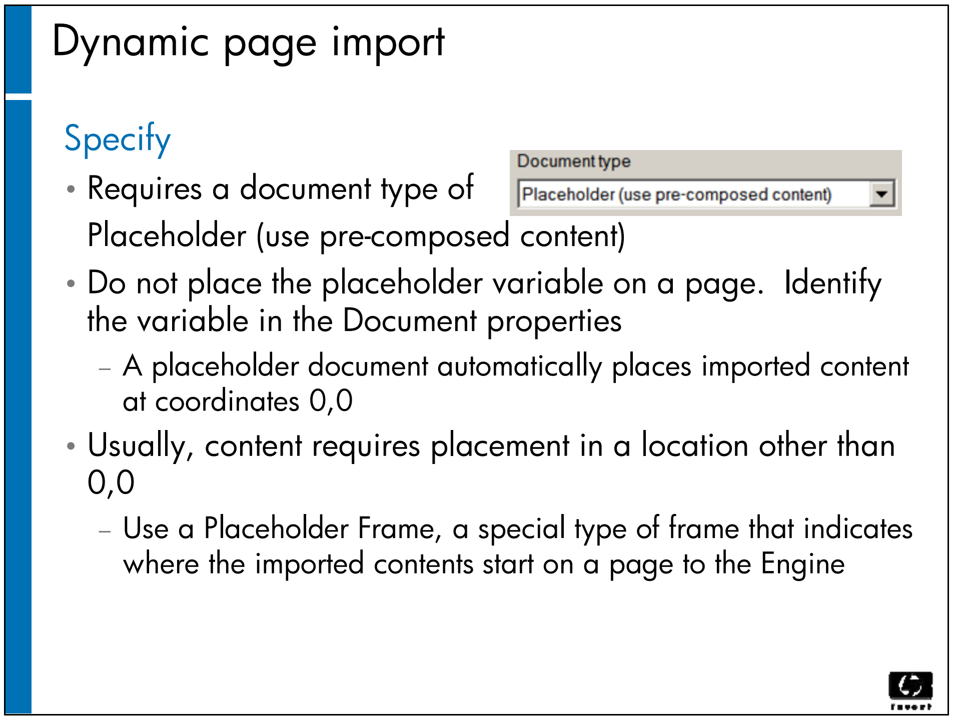
A similar feature to Dynamic Content Import, called Dynamic Page Import, enables you to import content to fill a page, multiple pages, or an entire document, with a single placeholder variable.



Dynamic page import

External content

- Uses placeholder variables and supports the same import types (RTF, B&W TIFF, and so forth) as Dynamic Content Import
- Import content for multiple pages, or an entire document
  - To enable, make the placeholder variable an array. Each element identifies a content source for a separate page
  - Design a page once and have the Engine use this page as a template to create as many pages ("filler pages") as needed. This is similar to the "Copy this Page" property
  - May specify a different end page design
  - The imported contents determine space taken up on a page



Dynamic page import

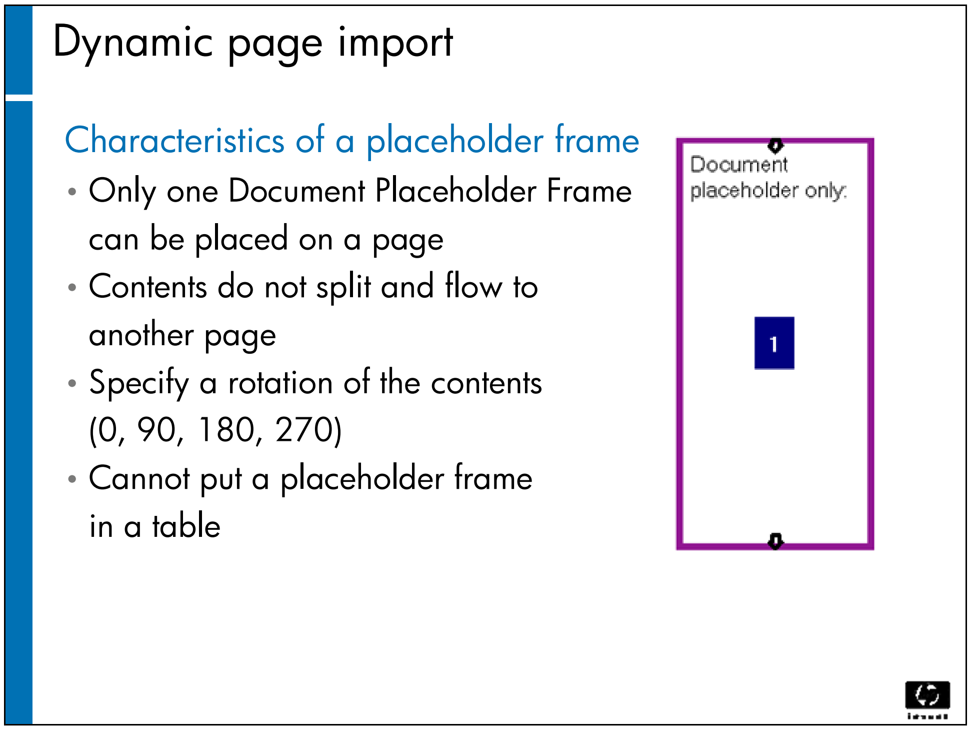
Specify

Document type

Placeholder (use pre-composed content)

- Requires a document type of Placeholder (use pre-composed content)
- Do not place the placeholder variable on a page. Identify the variable in the Document properties
  - A placeholder document automatically places imported content at coordinates 0,0
- Usually, content requires placement in a location other than 0,0
  - Use a Placeholder Frame, a special type of frame that indicates where the imported contents start on a page to the Engine

The purple Placeholder Frame holds a place for the incoming content on the pages used by the document. These frames can have borders like normal whitespace frames. Its upper left corner tells the Engine where to start the importing of external text/images.



Dynamic page import

Characteristics of a placeholder frame

- Only one Document Placeholder Frame can be placed on a page
- Contents do not split and flow to another page
- Specify a rotation of the contents (0, 90, 180, 270)
- Cannot put a placeholder frame in a table

Document placeholder only.
1

If you do not place a Placeholder Frame on a page, the Engine starts importing content at coordinates 0, 0.

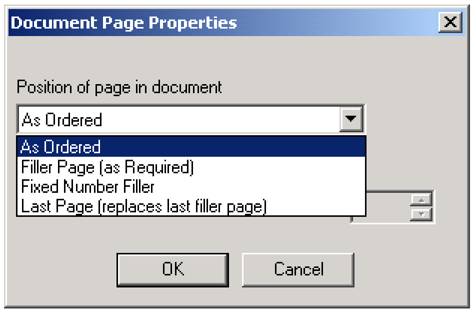
Specify Filler Pages in Edit Panel

There are **Position of Page in Document** options specific to placeholder documents. You specify *Filler Pages* when you have a document in the Edit Panel. This allows you to control the page positions in placeholder documents.

- After you have added pages to your document, drag it to the Edit Panel.
- Double-click the rightmost column of the document in the Edit Panel.

The Document Page Properties dialog box appears

Document Page Properties Dialog Box


Document Page Properties
Position of page in document
As Ordered
As Ordered
Filler Page (as Required)
Fixed Number Filler
Last Page (replaces last filler page)
OK
Cancel

- Select one of the following from the **Position of Page in Document** drop-down list:
  - **As Ordered –** This is the default option outside of Dynamic Page Import.
  - **Filler Page (as Required) –** Use this option to create as many pages as are needed for the document content, as long as there are no other restrictions on the number of pages in another option in Dialogue. One page will be created for each element in the array of the Placeholder variable that controls the document.
  - **Fixed Number Filler –** Enter the number of filler pages in the combo box after you make this choice. Only the specified number of pages will be created from the placeholder document.
  - **Last Page (replaces last filler page) –** Use this option to ensure this is the very last page in the application.
- Click **OK**.

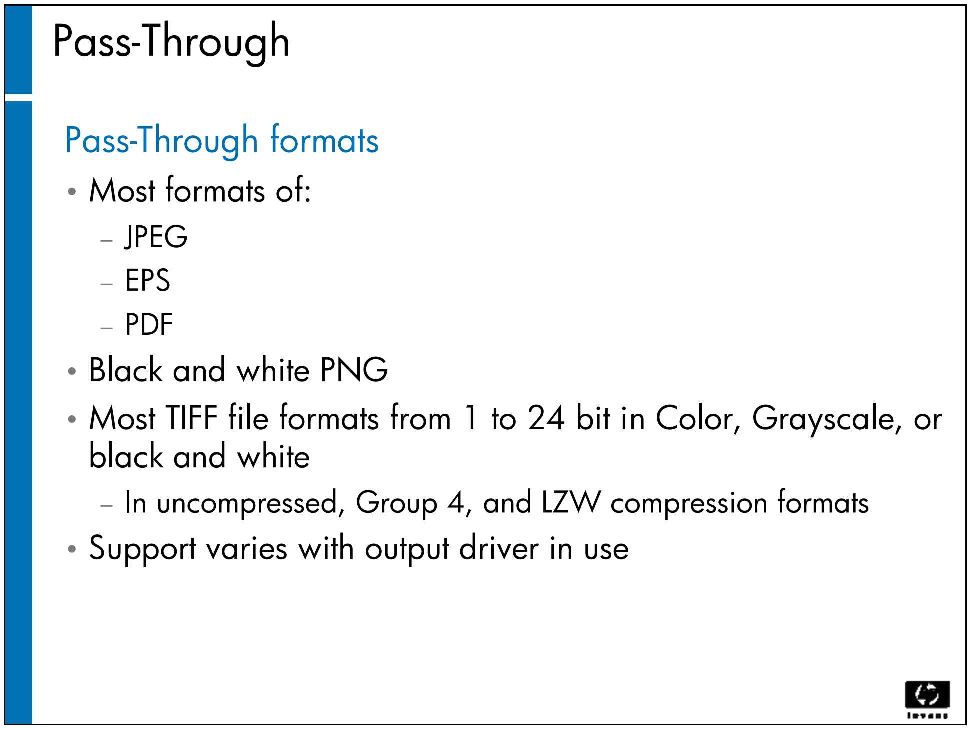
Additional information on **Dynamic Page Import** appears in the *302 :: Creating Long Documents* Dialogue training course.

12.7 Pass-Through

Placeholder Variables also support a **Pass-Through** feature for some imported file formats. The Engine handles pass-through files differently than the other formats in Dynamic Content Import.

- When you pass-through PDF or black and white TIFF pages, you can select a variable that specifies which pages to pass-through.
- When pass-through resources reach an output driver that does *not* natively support their format, they are ignored rather than converted to a suitable format for output.
- If the Engine processes a resource that *is* natively supported by an output driver, it inserts the original file – in its original format – into the print stream at run time.

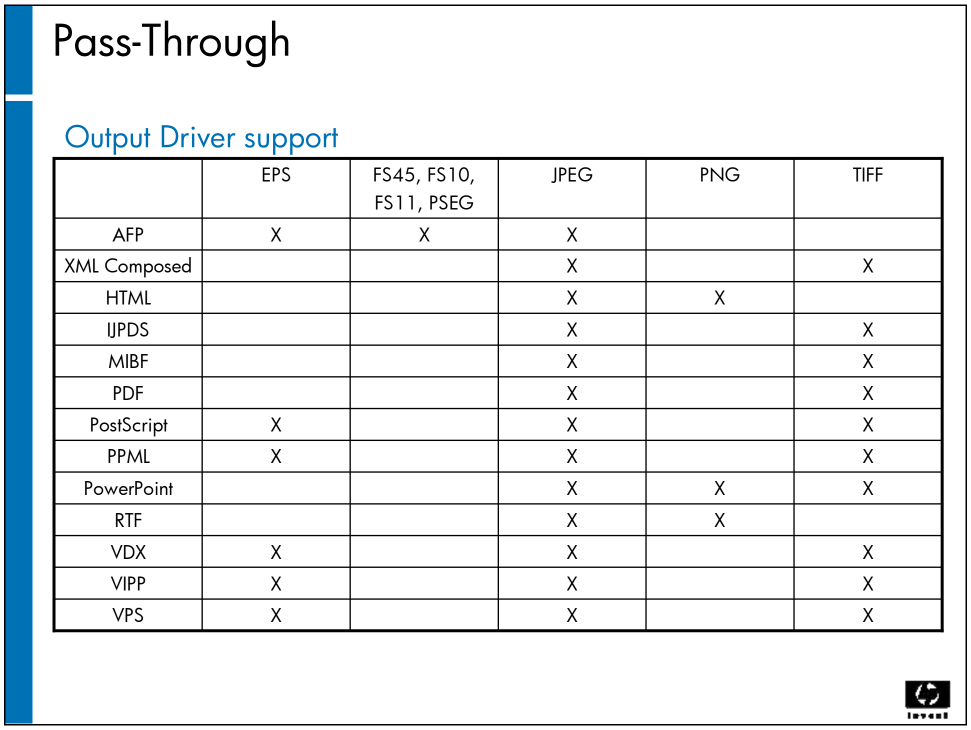
For more information regarding the Pass-Through feature please see the **Pass-Through Resources** section in the **Dynamically Importing Content** guide.



Pass-Through

Pass-Through formats

- Most formats of:
  - JPEG
  - EPS
  - PDF
- Black and white PNG
- Most TIFF file formats from 1 to 24 bit in Color, Grayscale, or black and white
  - In uncompressed, Group 4, and LZW compression formats
- Support varies with output driver in use



Pass-Through

Output Driver support

	EPS	FS45, FS10, FS11, PSEG	JPEG	PNG	TIFF
AFP	X	X	X		
XML Composed			X		X
HTML			X	X	
IJPDS			X		X
MIBF			X		X
PDF			X		X
PostScript	X		X		X
PPML	X		X		X
PowerPoint			X	X	X
RTF			X	X	
VDX	X		X		X
VIPP	X		X		X
VPS	X		X		X

Specifications

- Stripes are not supported in TIFFs of any kind. Most printers cannot handle stripes and extensive processing would be required to remove them.
- Customers using Black and White TIFFs should continue to use the existing B&W TIFF G4 placeholders for processing efficiency.
- If you dynamically import 1-bit or CCITT Group4 compressed TIFF files with multiple pages, you must specify the Placeholder variable as an array. Each element of the array is filled with one page, so you can specify groups or a series of pages to be imported.
- JPEG pass-through is limited to color images.
- EPS files must contain %%BoundingBox – this should not be an issue, as it is in almost all EPS files. You receive an error if it is not present.
- Dynamically imported PNG images are converted to raw bitmaps so they can be composed with any driver.
- The use of the -RESMANAGERFILE command is recommended. You will learn more about this file after the exercise.
- You can import large, multiple-page PDFs as objects into your design. You can specify a specific page, a range of pages, or all pages to be imported.

- You can optimize memory use when you dynamically import PDF content and use a resource management feature such as ResManager or the **Resource inclusion** drop-down menu. Use –SPLIT_LARGE_P.
- You can pass-through multiple PDF forms containing XFA script or page references. In addition, these forms can be placed at any location in the output (they do not have to be top-of-file), and they can be saved as any type of editable PDF form to be passed-through correctly.
- You can include editable PDF forms as pass-through objects to PDF output, with the forms still editable in the output. Editable PDF forms containing XFA script or page references can only be placed at the beginning of output. PDF 1.6 forms created with Adobe Designer must be saved as Adobe 6.0 Compatible Static PDF forms to be passed-through correctly. Also, you cannot resize or reposition any PDF pass-through objects when passing-through editable PDFs to an output using the AcroForm Pass-through feature.
- You can produce PDF/A-1b output, which is PDF output optimized for archiving.
- You can dynamically import PDF version 1.6 document

Chapter 13: Tag Sets

13.1 In this Chapter

This chapter details how to create customized tag sets, available with the Dynamic Content Import module.

Objectives

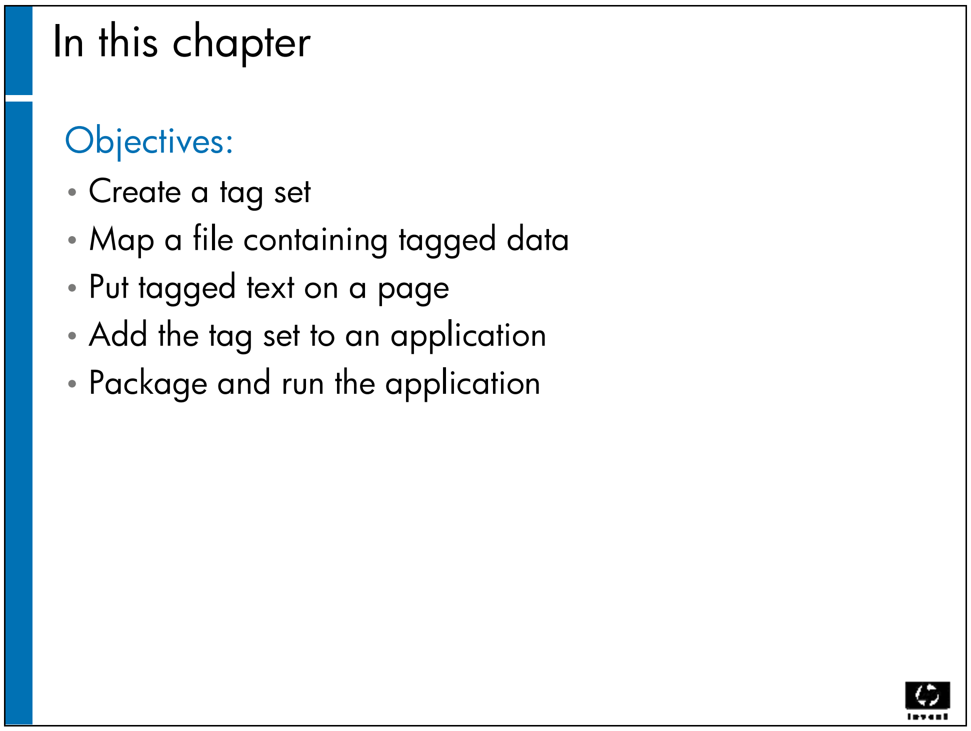
By the end of this chapter you will:

- Create a tag set.
- Map a file containing tagged data.
- Put tagged text on a page.
- Add the tag set to an application.
- Package and run the application.

For more information

For more information, refer to the following Dialogue documentation.

- Dynamically Importing Content guide Dynamic.pdf
- System Administration guide   System.pdf

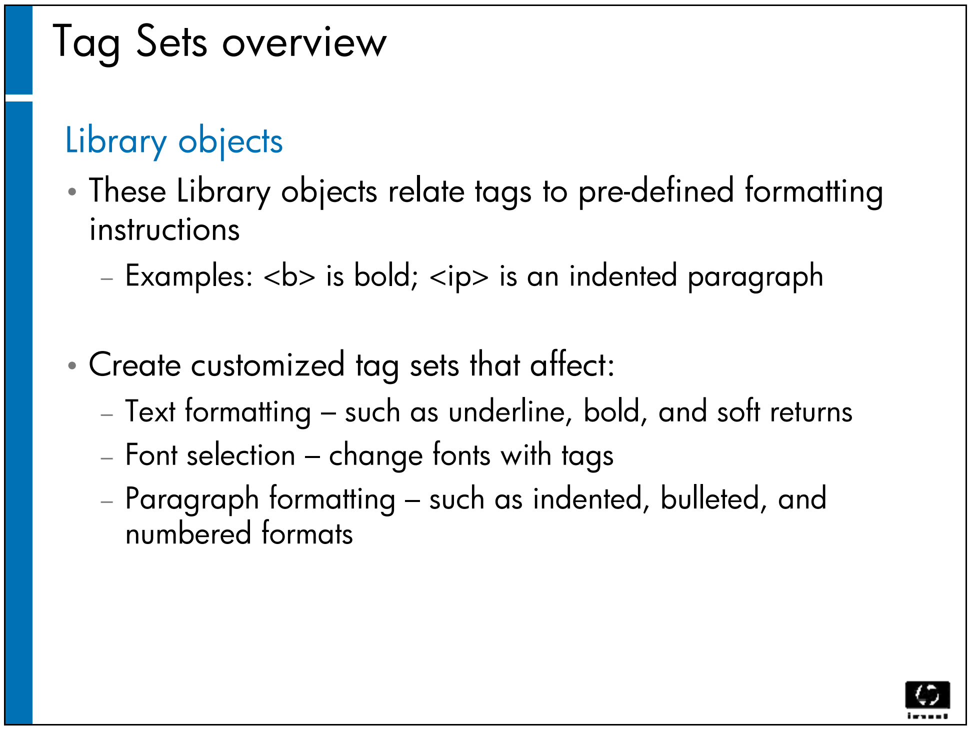


In this chapter

Objectives:

- Create a tag set
- Map a file containing tagged data
- Put tagged text on a page
- Add the tag set to an application
- Package and run the application

13.2 Tags Sets Overview



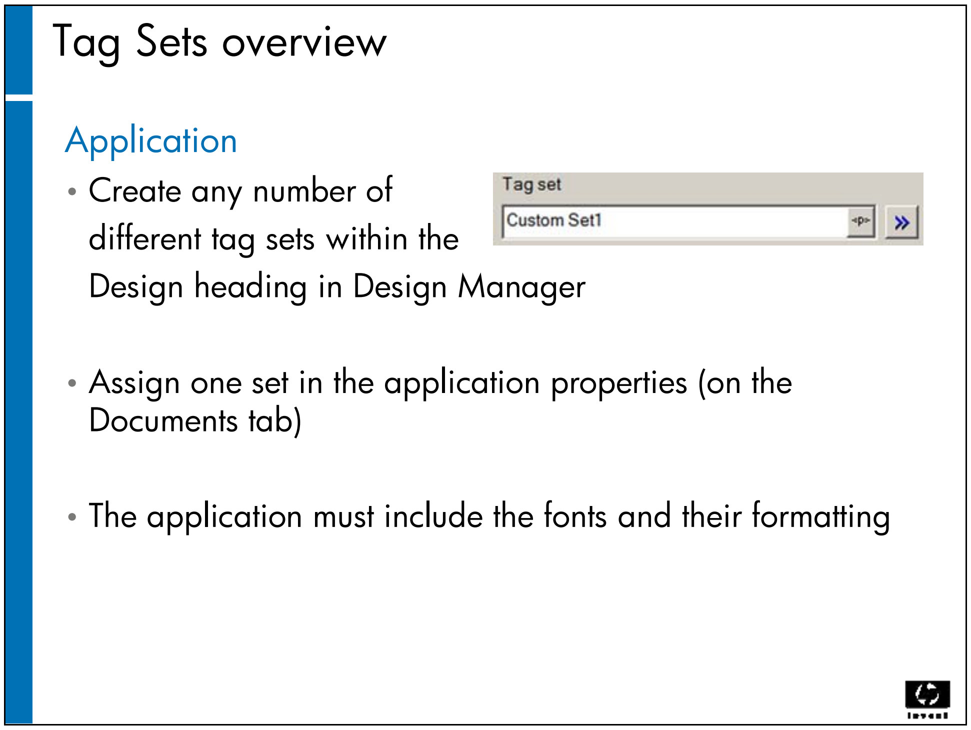
Tag Sets overview

Library objects

- These Library objects relate tags to pre-defined formatting instructions
  - Examples: <b> is bold; <ip> is an indented paragraph
- Create customized tag sets that affect:
  - Text formatting – such as underline, bold, and soft returns
  - Font selection – change fonts with tags
  - Paragraph formatting – such as indented, bulleted, and numbered formats

Tag Set objects reside in the Library, under **Environment** -> **Design**.

In order to use **Tag Sets**, you must have the **Dynamic Content Import** module enabled on your system.



Tag Sets overview

Application

- Create any number of different tag sets within the Design heading in Design Manager
- Assign one set in the application properties (on the Documents tab)
- The application must include the fonts and their formatting

Tag set: Custom Set1

In Designer, place Tag Set variables only in text boxes that have the **Autofit Text** drop-down list (in the **Text Box** tab) set to **None**. The other Autofit options do not support Tag Sets.

13.3 Define a Tag Set

Create New Tag Set Object

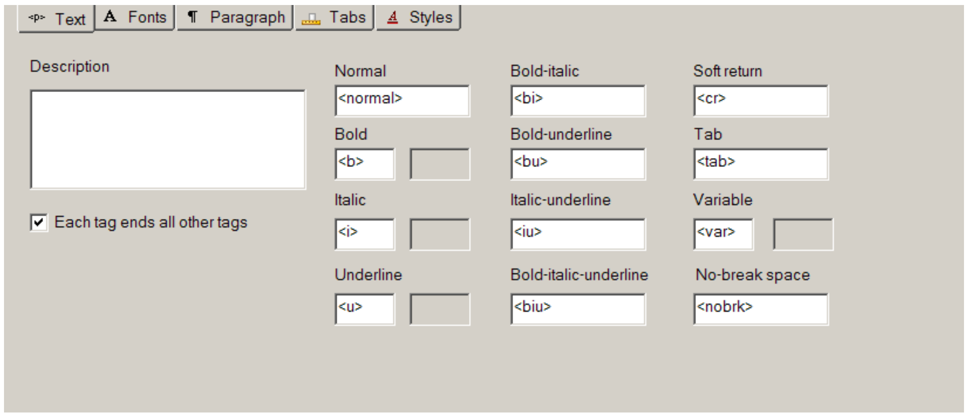
- Under **Environment** > **Design**, right-click the **Tag Sets** heading in the Library.
- Select **New Tag Set**.
  The **Name** dialog box appears.
- Enter a name in the **Name** dialog box and a description if you want.
- Click **Finish**.

It appears in the Property Panel. The Tag Set properties have three tabs: **Text**, **Fonts**, and **Paragraph**.

Text

For Dialogue to interpret the tags in your data, you must define each tag that varies from the defaults.

Tag Set Properties -Text Tab



Description

☑ Each tag ends all other tags

Normal	`<normal>`	Bold-italic	`<bi>`
Bold	`<b>`	Bold-underline	`<bu>`
Italic	`<i>`	Italic-underline	`<iu>`
Underline	`<u>`	Bold-italic-underline	`<biu>`
Soft return	`<cr>`	Tab	`<tab>`
Variable	`<var>`	No-break space	`<nobrk>`

End Tags

By default, the **Each tag ends all other tags** check box is selected. With this setting, the end of one tag is signified by the beginning of the next. Since end tags are not needed, the end tag properties on this tab are dimmed.

You can enforce the use of end tags.

- Clear the **Each tag ends all other tags** check box (available on all tabs).

The end tag properties on the **Text** tab become available.

- Make any updates to the default end tags as needed.

Fonts

You can change fonts in the flow of your text with font tags. Dialogue automatically packages the fonts referenced by these tags when you include the tag set in an application.

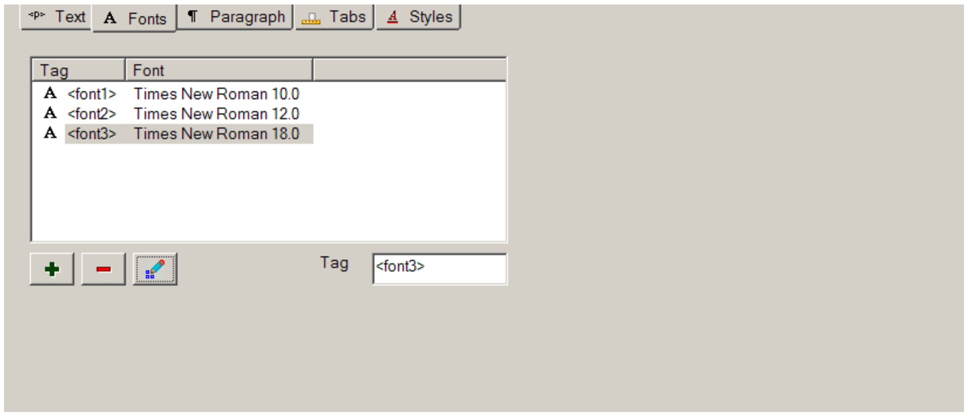
Add Font Tag

- Click the **Fonts** tab.
- Click the **Plus** (+) button .

Dialogue will:

- Create a tag under the **Tag** column in the text box area. The default naming convention is <font1>, <font2>, and so on. A capital letter **F** identifies this item as a font tag.

Default Font Tags in Fonts Tab



Tag	Font
<font1>	Times New Roman 10.0
<font2>	Times New Roman 12.0
<font3>	Times New Roman 18.0

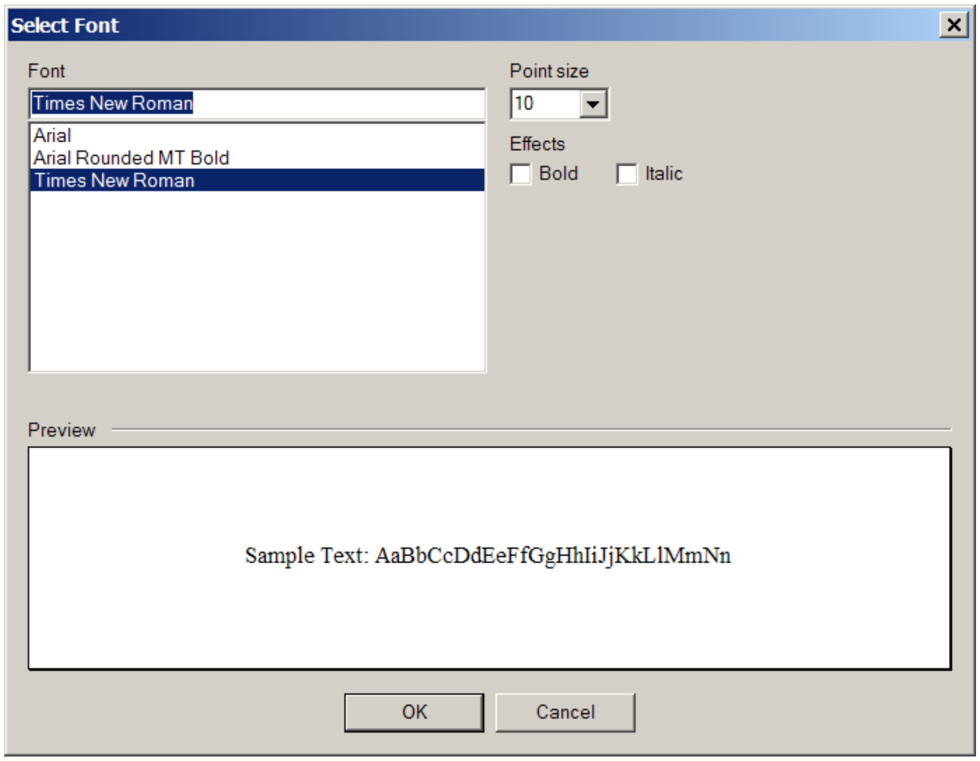
Tag: <font3>

Edit Font Tag

You can change the tag text or the font information for any tag that appears on this tab.

- Highlight the font tag you want to change.
  The **Tag** text box, button, and button become available.
- To edit the tag text, type your change into the **Tag** text box. Be sure to include brackets.
  Changes you make in the **Tag** text box are immediate.
- To edit the font information, press the button.
  The **Select Font** dialog box appears.

Select Font dialog box



Font: Times New Roman (selected from the list: Arial, Arial Rounded MT Bold, Times New Roman)

Point size: 10

Effects: Bold [ ] Italic [ ]

Sample Text: AaBbCcDdEeFfGgHhIiJjKkLlMmNn

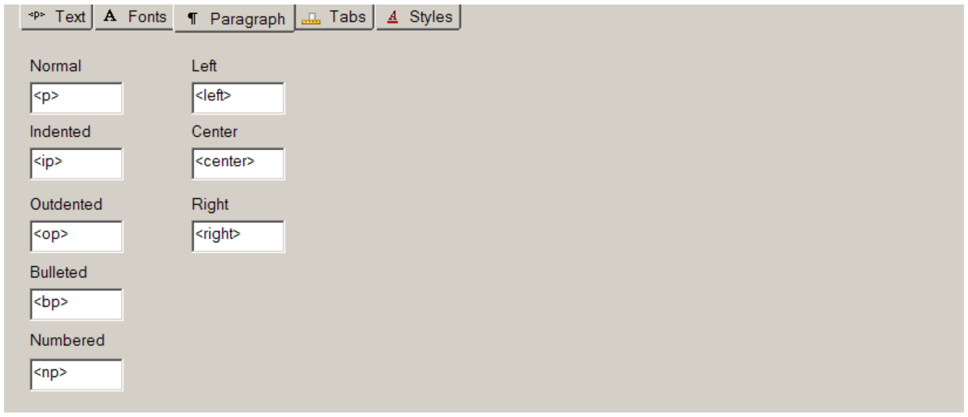
- Make your changes to this dialog box as needed.
  - To change a font, choose a new font from the top drop-down list.
  - To change the point size, choose a new size from the **Point Size** drop-down list.
  - You can set the font to always appear bold, italic, or both with the **Bold** and **Italic** check boxes.
- Click **OK**.

You return to the Tag Set properties in the Property Panel with your changes under the **Font** column.

Paragraph

The tags in this tab affect the formatting of paragraphs. The first column of options affects the layout style of the paragraph. The second column of options contains alignment settings for all text in the paragraph.

Tag Set Properties - Paragraph Tab

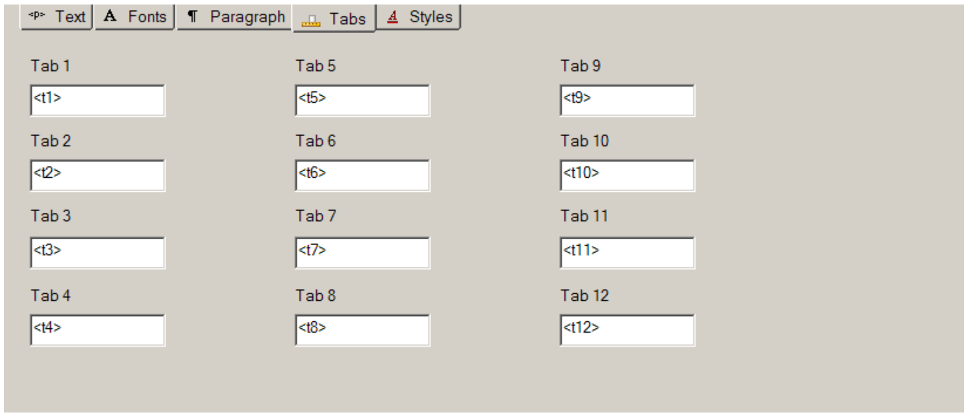


Normal: `<p>`
Indented: `<ip>`
Outdented: `<op>`
Bulleted: `<bp>`
Numbered: `<np>`
Left: `<left>`
Center: `<center>`
Right: `<right>`

Tabs

Tagged text can be positioned to specific tab positions. You define the tags for up to twelve tab stops (in left to right order).

Tag Set Properties - Tabs Tab



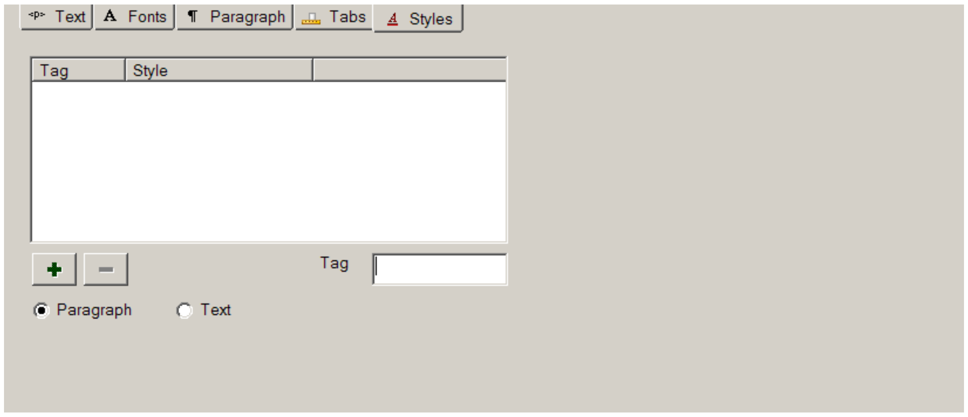
Tab 1: `<t1>`
Tab 2: `<t2>`
Tab 3: `<t3>`
Tab 4: `<t4>`
Tab 5: `<t5>`
Tab 6: `<t6>`
Tab 7: `<t7>`
Tab 8: `<t8>`
Tab 9: `<t9>`
Tab 10: `<t10>`
Tab 11: `<t11>`
Tab 12: `<t12>`

You set the exact tab locations (and tab type) using the Paragraph Format Ruler in Designer. Tab 1 in the *Tabs* tab refers to the leftmost user-defined tab in the Ruler.

Styles

You can apply styles to tags, provided you specify the style on a style sheet and you assign the style sheet to the design page.

Tag Set Properties - Styles Tab

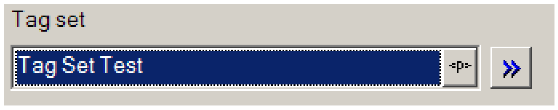


The *Paragraph* and *Text* radio buttons at the bottom of the Property Panel control the tags you can see. If you click the *Paragraph* radio button, only tags for Paragraph styles are shown in the text box. Likewise, if you click the *Text* radio button, only tags for Text styles are shown in the text box.

In addition, when you add a tag, the styles available in the *Select Style* dialog box are filtered based on the current radio button selected.

Applications and Tag Sets

Application Properties – Document Tab



Tag set

Tag Set Test

You select the tag set to use for an application on the **Document** tab.

When you use your **Tagged Text Variable** on an object, you simply insert the variable or variables you have defined into a designed object.

13.4 Exercise: Use Tag Sets

In this exercise, you create a tag set to define the formatting of imported text. Text formatting includes bolding, italics, underline, font face, and font size changes.

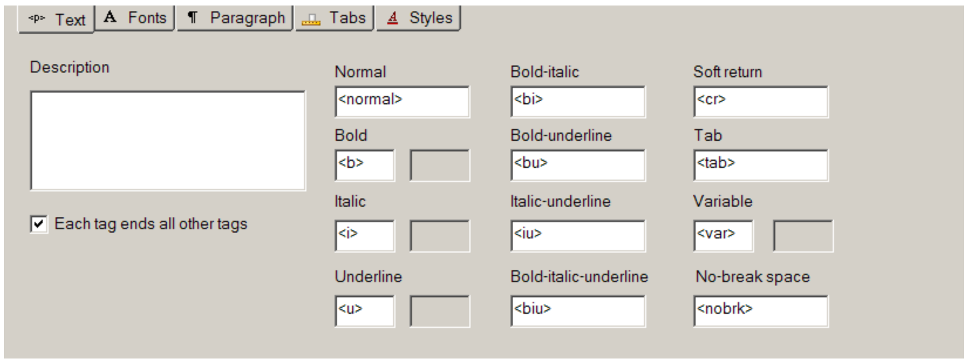
A. Create a New Tag Set

1. In the Library, expand the **Environment** and **Design** headings.
2. Right-click the **Tag Sets** heading.
3. Select **New Tag Set** from the shortcut menu.

The **Name** dialog box appears.

4. Enter `Class 210 Tag Set` in the **Name** dialog box.
5. Click **Finish**.
   The tag set appears in the Property Panel for you to define.
6. Clear the **Each tag ends all other tags** check box on the left side of the dialog box.
7. Leave the other properties at their default settings.

Text Tab



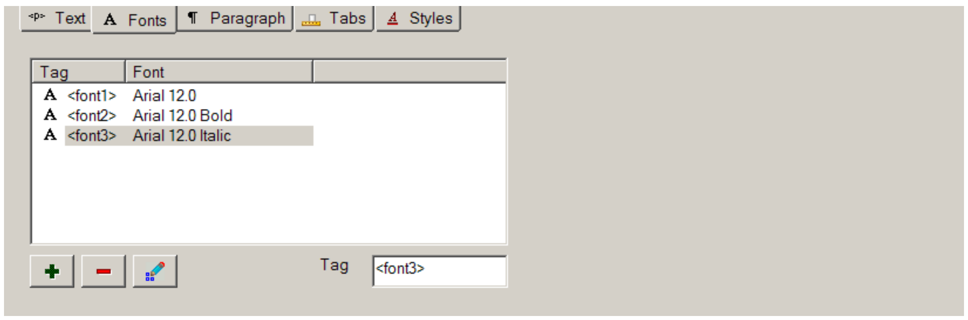
Description

[✓] Each tag ends all other tags

Normal: `<normal>`	Bold-italic: `<bi>`	Soft return: `<cr>`
Bold: `<b>`	Bold-underline: `<bu>`	Tab: `<tab>`
Italic: `<i>`	Italic-underline: `<iu>`	Variable: `<var>`
Underline: `<u>`	Bold-italic-underline: `<biu>`	No-break space: `<nobrk>`

8. Click the **Fonts** tab.
9. Add Arial 12, Arial 12 bold, and Arial 12 italic to the font set.

Fonts Tab



Tag	Font
`<font1>`	Arial 12.0
`<font2>`	Arial 12.0 Bold
`<font3>`	Arial 12.0 Italic

Tag `<font3>`

10. Save and close the tag set.

NOTE: For this exercise, no settings are configured on the **Paragraph**, **Tabs**, or **Styles** tabs.

B. Define Tagged Text Variables

1. Expand the Exercise 12-13 Dynamic Content Import folder.
2. Right-click the Data Dictionary and create a new variable:
   a. **Name:** TagsMessage
   b. **Type:** Tagged Text
   c. **Source:** `File only`

C. Create and Map the Data File

1. Expand the Exercise 12-13 Dynamic Content Import folder.
2. Right-click the Data Files heading and create a new data file:
   a. **Name**: Tags Reference
   b. **File type**: Reference file
   c. **File format**: Delimited data file
   d. **Key**: CustomerAccount
   e. **Delimiter**: "," (comma)
   f. Check the **Fields may be 'quoted' with** checkbox.
   g. **Test data source**: C:\210 Advanced Data Concepts\Data Files\Tag Reference.dat
   h. **Production data source**: EXtags
   i. Leave the other properties at their defaults.
3. Save and close the Property Panel.
4. Drag the **Tags Reference** data file into the Edit panel.
5. Map the first data area to the **CustomerAccount** variable in the **Exercise 12-13 Dynamic Content Import** folder.
6. Map the second data area to the **TagsMessage** variable you created earlier. When the **Data Area Properties** dialog box appears, type `500` in the **Length** field.
7. Save and close the Edit Panel.

8. Drag the **Tags Reference** data file to the **Tags Application**.

NOTE: Ensure the order of the data files, from top to bottom, is as follows: Tags Customer Driver File and Tags Reference File.

D. Insert Variable on the Tags Page

1. Open the Tags Page from the Exercise 12-13 Dynamic Content Import folder in Designer.
2. Replace the red 'TagsMessage' text with the TagsMessage variable from the Exercise 12-13 Dynamic Content Import folder.
3. Save the page and exit Designer.

E. Define the Tag Set for the Application

1. Drag the Tags Application from the Exercise 12-13 Dynamic Content Import folder to the Property Panel.
2. Click the Documents tab.
3. From the Tag Set list, choose the Class 210 Tag Set you created earlier.
4. Save and close the Property Panel.

F. Package and Run the Engine

1. Package and run the application.
2. Review the results in the Exstream Viewer.

Chapter 14: XML Input

14.1 In this Chapter

Dialogue supports data source files written in XML format. This optional XML Input module provides the capabilities detailed in this chapter.

Objectives

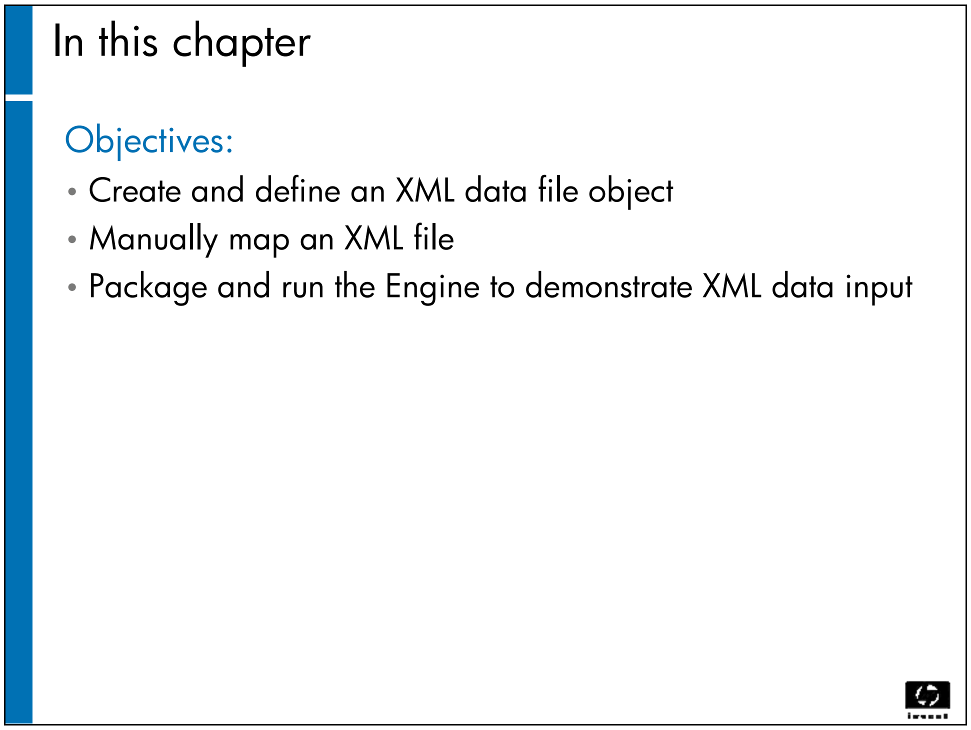
By the end of this chapter you will:

- Create and define an XML data file object.
- Manually map an XML file.
- Package and Run the Engine to demonstrate XML data input.

For more information

For more information, refer to the following Dialogue documentation.

- *Dynamic* Data Access guide DDA.pdf

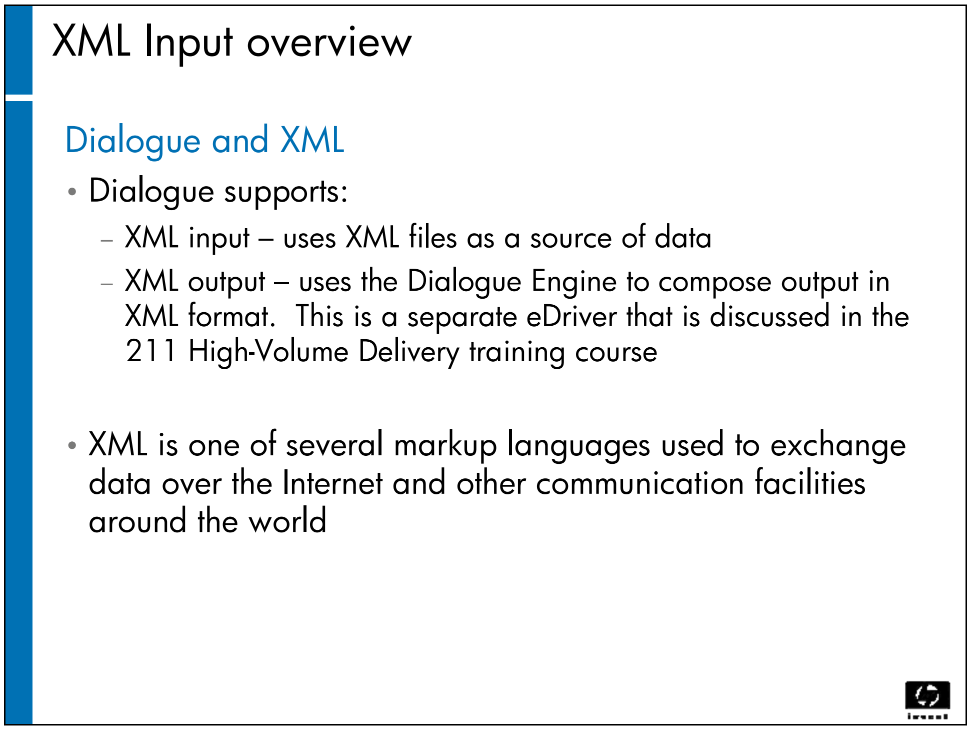


In this chapter

Objectives:

- Create and define an XML data file object
- Manually map an XML file
- Package and run the Engine to demonstrate XML data input

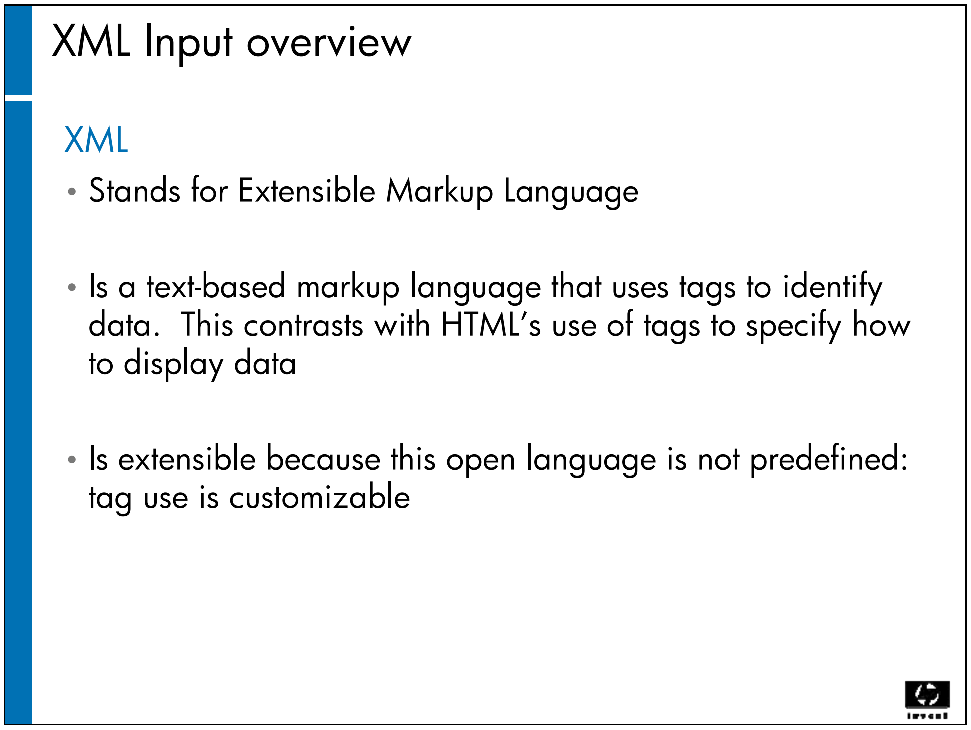
14.2 XML Input Overview



XML Input overview

Dialogue and XML

- Dialogue supports:
  - XML input – uses XML files as a source of data
  - XML output – uses the Dialogue Engine to compose output in XML format. This is a separate eDriver that is discussed in the 211 High-Volume Delivery training course
- XML is one of several markup languages used to exchange data over the Internet and other communication facilities around the world

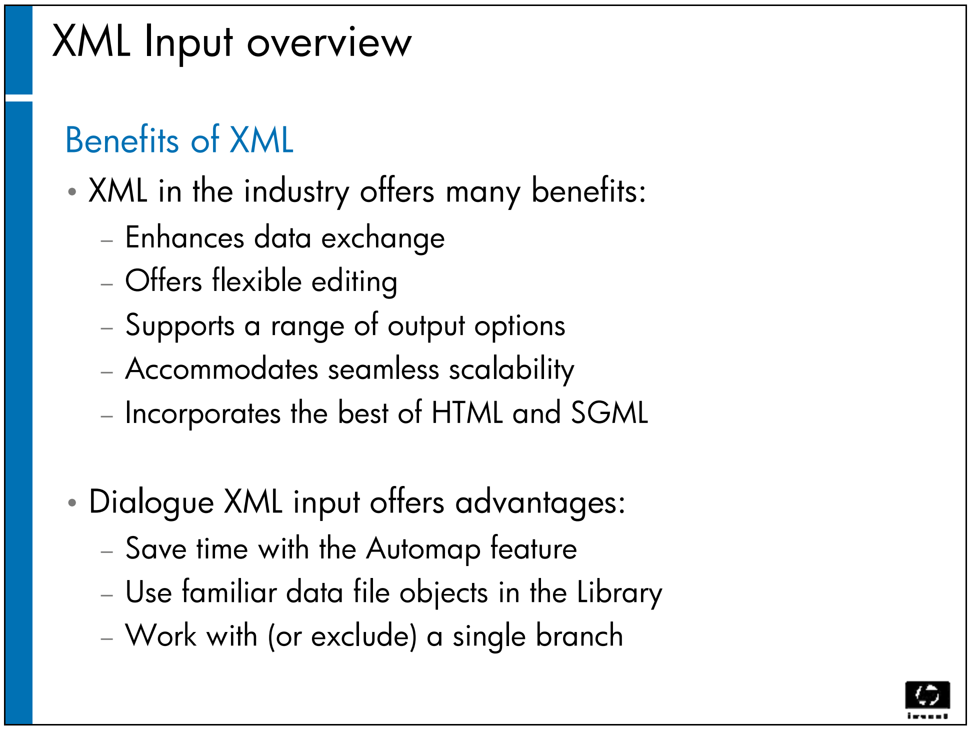


XML Input overview

XML

- Stands for Extensible Markup Language
- Is a text-based markup language that uses tags to identify data. This contrasts with HTML's use of tags to specify how to display data
- Is extensible because this open language is not predefined: tag use is customizable

Benefits of XML



XML Input overview

Benefits of XML

- XML in the industry offers many benefits:
  - Enhances data exchange
  - Offers flexible editing
  - Supports a range of output options
  - Accommodates seamless scalability
  - Incorporates the best of HTML and SGML
- Dialogue XML input offers advantages:
  - Save time with the Automap feature
  - Use familiar data file objects in the Library
  - Work with (or exclude) a single branch

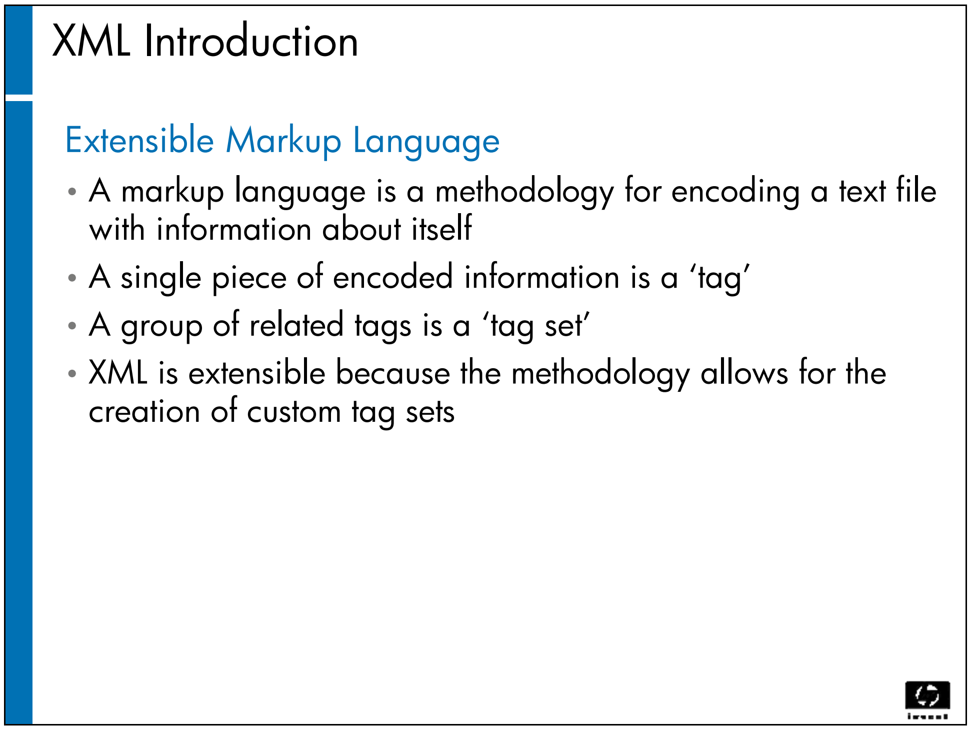
Advantages of XML in the industry:

- **Enhances data exchange**. Flexible ways to identify information, XML improves the functionality of the Web – such as exact searches. The different segments of the data are identified, various multi-vendor applications use the data in different ways.
- **Offers flexible editing**. Create and edit files using anything from a simple text editor to a complex graphical interface program. Its use of predictable and consistent notation helps make these files easy to update and debug.
- **Supports a range of output options**. Since you define how the data will be used, you can tailor the output for a wide variety of formats, including future formats. An additional advantage is XML's ability to let you modularize your documents.
- **Accommodates seamless scalability**. The efficiency and predictability of XML code makes this language adaptable from a small to very large files.
- **Incorporates the best of HTML and SGML**. Create your own customized markup applications, XML takes the best of the inflexible HyperText Markup Language (HTML) and the more flexible, but more complex, Standard Generalized Markup Language (SGML) to offer a robust, predictable solution to the exchange of data on and off the Internet.

Advantages of Dialogue XML input:

- **You can save time with the *Auto-Map* feature**. Makes mapping and re-mapping faster than manual methods.
- **You use familiar data file objects in the Library** – such as Customer Driver Files, Initialization Files, or Reference Files.
- Since Dialogue recognizes the hierarchical structure of XML files, **you can work with (or exclude) a single branch** of a file's hierarchy.

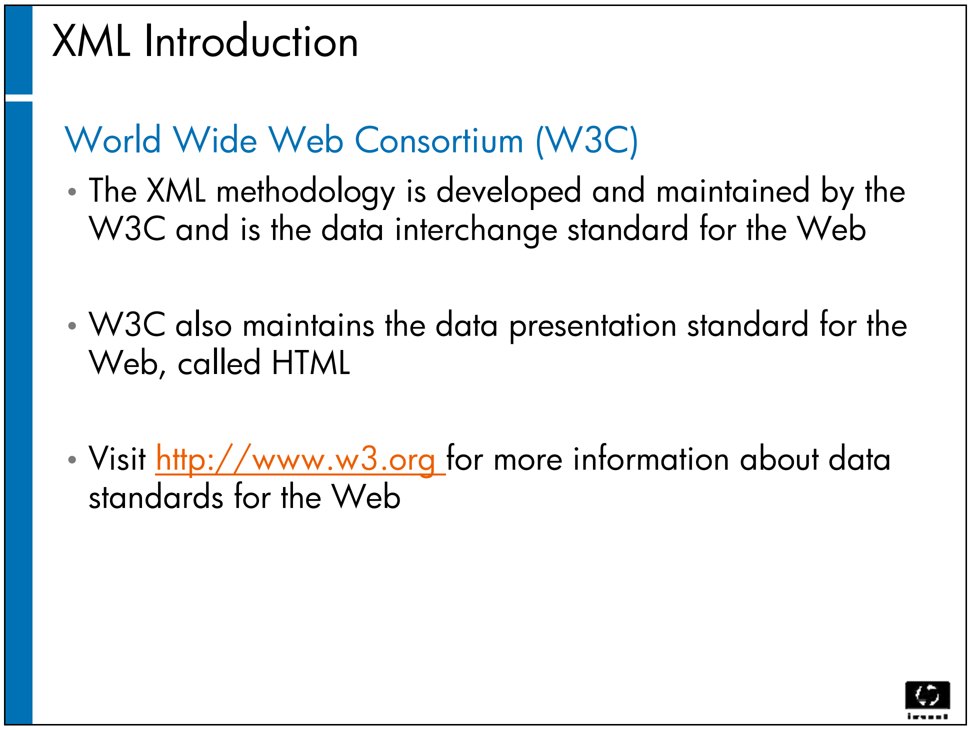
14.3 XML Introduction



XML Introduction

Extensible Markup Language

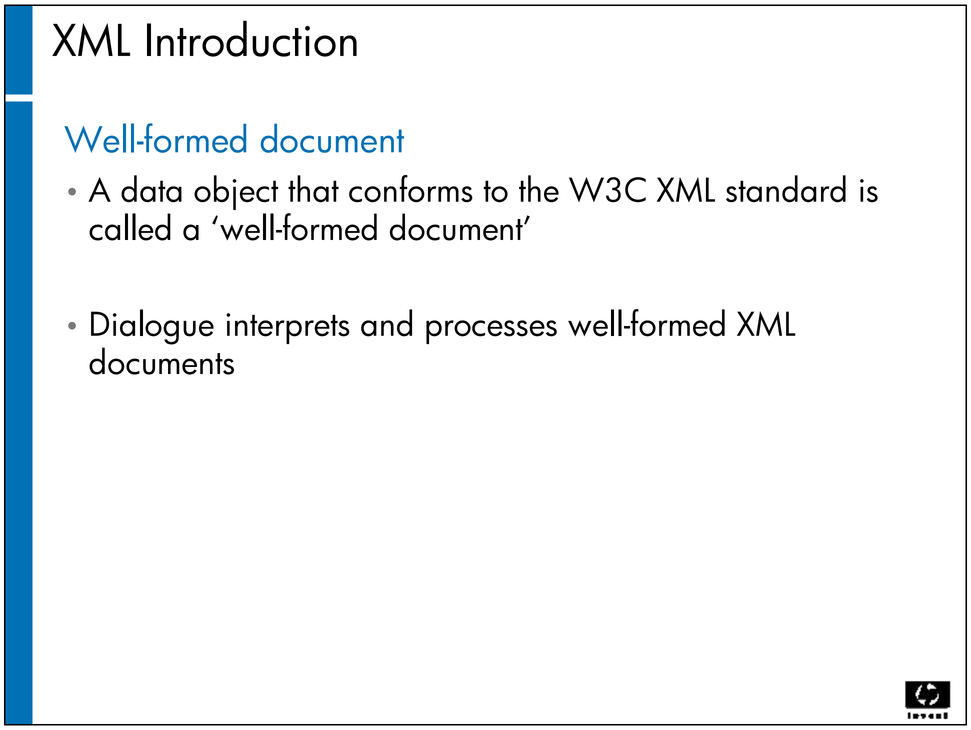
- A markup language is a methodology for encoding a text file with information about itself
- A single piece of encoded information is a 'tag'
- A group of related tags is a 'tag set'
- XML is extensible because the methodology allows for the creation of custom tag sets



XML Introduction

World Wide Web Consortium (W3C)

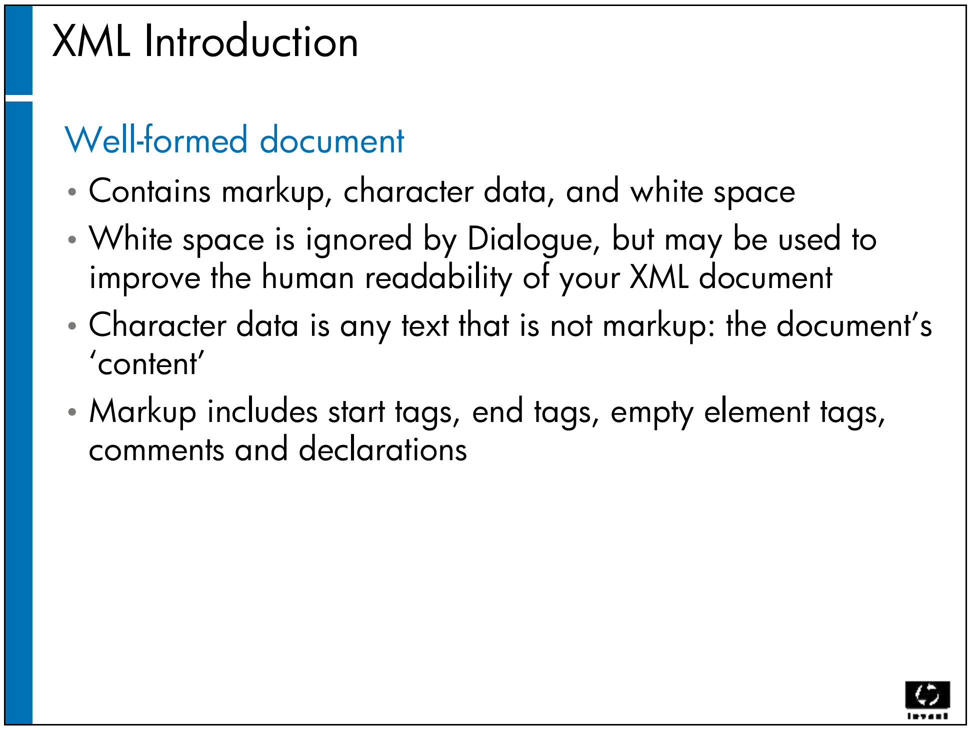
- The XML methodology is developed and maintained by the W3C and is the data interchange standard for the Web
- W3C also maintains the data presentation standard for the Web, called HTML
- Visit http://www.w3.org for more information about data standards for the Web



XML Introduction

Well-formed document

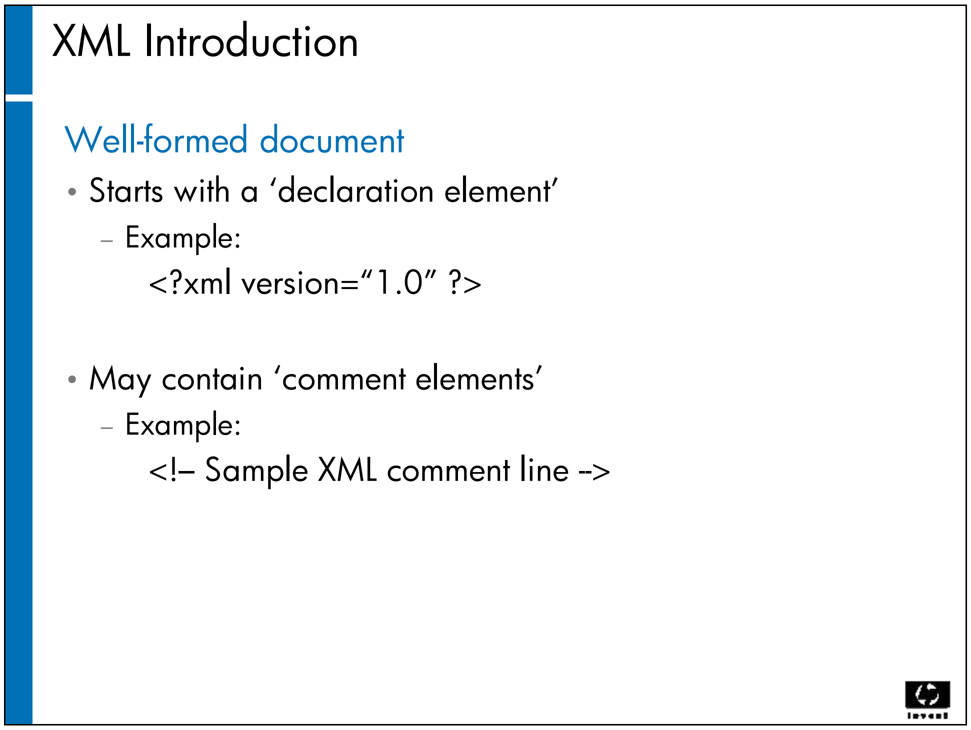
- A data object that conforms to the W3C XML standard is called a 'well-formed document'
- Dialogue interprets and processes well-formed XML documents



XML Introduction

Well-formed document

- Contains markup, character data, and white space
- White space is ignored by Dialogue, but may be used to improve the human readability of your XML document
- Character data is any text that is not markup: the document's 'content'
- Markup includes start tags, end tags, empty element tags, comments and declarations



XML Introduction

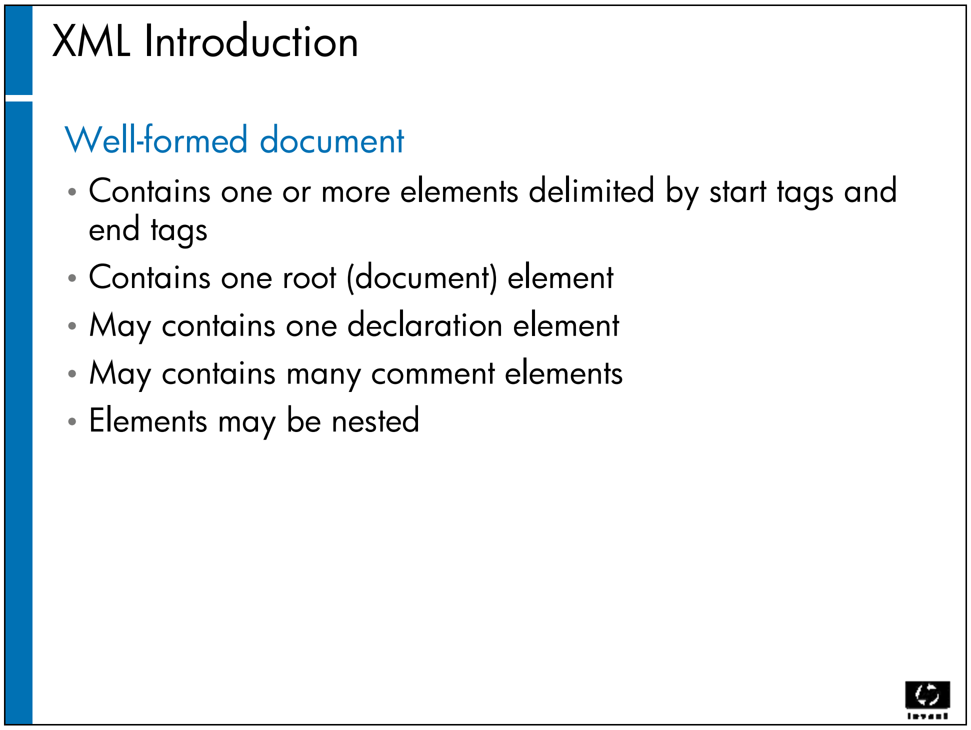
Well-formed document

- Starts with a ‘declaration element’
  - Example:

    <?xml version="1.0" ?>

- May contain ‘comment elements’
  - Example:

    <!– Sample XML comment line ->



XML Introduction

Well-formed document

- Contains one or more elements delimited by start tags and end tags
- Contains one root (document) element
- May contains one declaration element
- May contains many comment elements
- Elements may be nested

14.4 A look at XML Code

Compare XML with HTML

At first glance, XML may look similar to HTML: both use tags and ignore any whitespace. However, there are major differences between the two languages.

- XML tags *label* data. HMTL tags specify how to *display* data.
- HTML is a pre-defined language for a single, inflexible document type. You customize how you assign tags in XML to produce user-defined document types.
- *Well-formed* XML follows stricter rules than HTML.
- Quotes are optional for attribute values in HTML, but mandatory in XML.
- Commas between attributes in HTML are ignored. Commas between attributes in XML generate errors.

XML tags are said to act like field names in a program. You can assign any XML tags you want that make sense. In the following message, notice *none* of the tags tell a program how to display text or data:

```
<?xml version="1.0"?>
    <message>
          <to>mburns@bigware.com</to>
          <from>you@branchoffice.net</from>
          <subject>new insurance rates</subject>
          <text>
                Effective next week, the premium for smokers
                increases five percent.
          </text>
    </message>
```

Note each start tag has a corresponding end tag. Between the tags for <message> are numerous other tags. Since XML supports nested tags, it easily forms a hierarchical data structure.

XML Elements

Since XML ignores whitespace, you can generally position XML elements as needed for readability.

Declaration Element

XML files start with a **declaration element**, also called an **XML Prolog**, that tells the browser or program how to process the data in the file. The statement begins and ends with `<?` and `?>`.

Example:
```
<?xml version="1.0"?>
```

Comment Elements

XML supports comments if the text lies between `<!--` and `-->`.

Example:
```
<!--Mapping for XML for Transactions-->
```

Root Element

The **root element**, also called the **document element**, is simply the first tag after the declaration. All other tags must be nested within it. Like all other tags, it must have a companion end tag.

```
Example:
<Statement_Application>

(All nested tags)

</Statement_Application>
```

Components of a Tag

Tags can have four parts:

- Start tag
- Tag value
- Attributes – additional information included as part of the tag itself
- End tag

In this example, four parts are present:

```
<CustomerNumber Id="1">Robert Stevens</CustomerNumber>
```

In the last example, <CustomerNumber …> is the start tag, </CustomerNumber> is the end tag, Robert Stevens is the tag value, and Id="1" is an attribute (and attribute value) of the customer tag.

Character Codes

Some character codes that are legal in HTML, such as &copyr, are not recognized in XML. Standard XML has only five predefined entities, all supported in Dialogue:

- &gt   instead of   >

■ &lt	instead of	<
■ &apos	instead of	‘
■ &amp	instead of	&
■ &quot	instead of	"

To use other character codes you can either:

- Wrap the data with the appropriate characters in a CDATA (character data) record, which will not transpose the characters, or
- Use the following numeric values
  - &#160 no-break space (non-breaking space)
  - &#161 inverted exclamation mark
  - &#162 cent sign
  - &#163 pound sign
  - &#164 currency sign
  - &#165 ying / yang sign
  - &#166 broken bar
  - &#167 section sign
  - &#168 spacing dieresis
  - &#169 copyright sign

A Few Basic Constraints

Since more and more XML files are being generated by XML creation programs, adherence to constraints is becoming less of a manual task. For this reason, the following is only a sampling of the most basic constraints.

- Each start tag must have an end tag. An end tag has the same name as the start tag but is preceded with the “/” character. Examples:
  ```
  <Customer></Customer>
  <Date></Date>
  ```
- Every element requires separate start and end tags with the same spelling and upper/lower case. Example:
  ```
  <date>…</Date> is invalid
  ```
- All tags must nest correctly under higher-order elements with no overlaps.
- Tags without any tag values (“empty tags”) can be shown in simplified form. (You can use an empty tag to define a hierarchy in the nesting scheme, much like data sections in Dialogue.) Example: `<Date></Date>` can be simplified as

```
<Date/>. If it has one or more attributes it becomes:
     <Date="Publish Date"/>
```

- Attributes must have a name followed by the "=" sign followed by the attribute value in quotes. Attribute values without quotes are invalid. (This is allowed in `HTML, but not with XML.) An example of multiple attributes:`

```
     <Date="Publish Date" Run="Test" Operator="14">
```

Validating XML

Browser

XML documents can be viewed using a Web browser (preferably as late a version as possible), but this method is of limited use to alerting you to errors.

Parser

A better solution is a *parser*. This third-party software checks for errors and may offer recommendations for correction. A number of different kinds of XML file parsers are available in the global marketplace.

Dialogue

Dialogue conducts its own checks of XML files for use with the Dialogue Engine. When you drag an XML data file object from the Library to the Edit Panel, Dialogue's built-in parser displays messages in an **XML Messages** dialog box.

Visual Indicators in the Edit Panel

A *XML data file* object in the Library identifies a file in your system that is a well-formed XML file. Generally, this is a data source file that has an .xml extension.

When you drag a XML data file to the Edit Panel, Dialogue immediately maps tags values (you manually map tag names and attributes). The software automatically:

- Finds > and < characters in the file
- Places a carriage return between them
- Processes the tags, assigning different colors to tags and tag values

Text Color

Colors on text in the Edit Panel offer a quick indication of tags, values, and attributes in the XML file.

- Black Text – tag
- Red Text – attribute
- Blue Text – attribute value
- Green Text – data value

Background Color

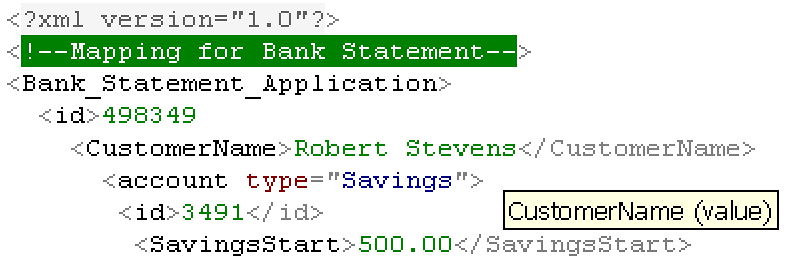
Colors *behind* text also have meaning.

- **White** denotes tags.
- **Pale Yellow** denotes an area that Dialogue recognizes as a tag.
- **Yellow** denotes a selected area in XML mapping.
- **Teal** denotes a mapped area.
- **Cyan** indicates locations where data areas overlap.
- **Black** denotes an area you have marked as **Ignore**.
- **Dark Green** denotes an area that is a comment and will be ignored by the Engine.
- **Near White** (a very pale gray) denotes the declaration statement. This color also identifies **CDATA** (*character data*, such as symbols, that are interpreted literally and not processed).

Pop-ups

When you place your cursor over an element, tag, attribute, or value in the XML file, Dialogue displays a pop-up display on the current information it has about the area.

Tag and Pop-Up

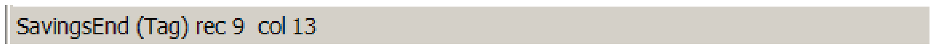


```
<?xml version="1.0"?>
<!--Mapping for Bank Statement-->
<Bank_Statement_Application>
  <id>498349
    <CustomerName>Robert Stevens</CustomerName>
      <account type="Savings">
       <id>3491</id>
        <SavingsStart>500.00</SavingsStart>
```

Line Indicators

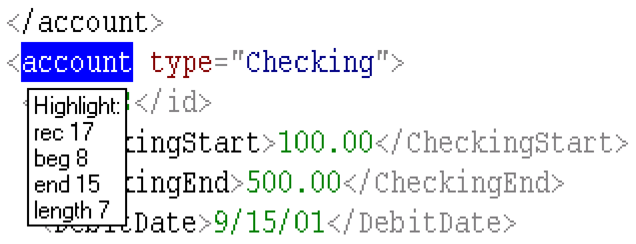
You can see the line (record) numbers in the **Status Bar** at the bottom of your window. You can also see the line numbers in the **pop-ups**.

Status Bar



SavingsEnd (Tag) rec 9 col 13

Pop-up



```
</account>
<account type="Checking">
 <        </id>
 <   ingStart>100.00</CheckingStart>
 <   ingEnd>500.00</CheckingEnd>
 <DebitDate>9/15/01</DebitDate>
```

14.5 Exercise: Map and Use XML Input

The Auto-Map feature reduces the time it takes to map and use an XML data file. However, it is not as flexible as the manual method used in this exercise. If you need complete control of variable properties, use manual mapping.

A. Create the XML Data File Object

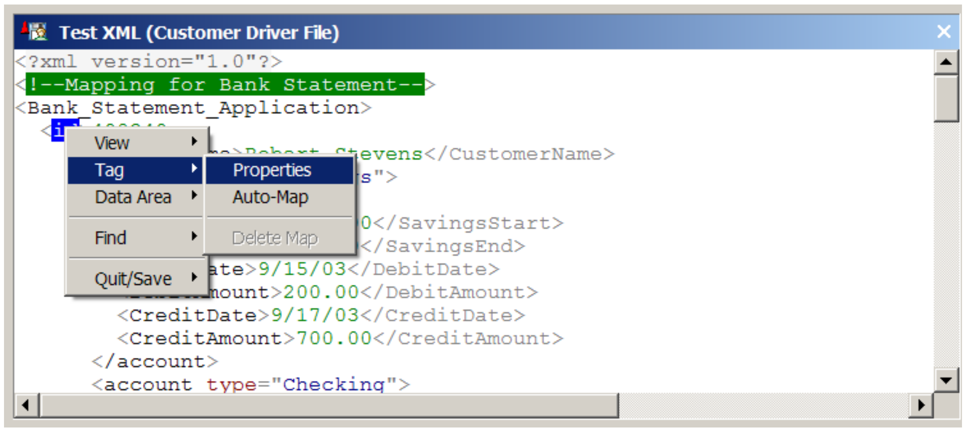
You create a new XML data file as you do other data file objects in Dialogue.

1. In the Library, expand the **Exercise 14 XML Input** folder.
2. Right-click the **Data Files** heading.
3. Choose **New Data File** from the shortcut menu.
   The **New Data File** dialog box appears.
4. Type `XML Mapping` in the **Name** text box.
5. Choose a **File Type** of **Customer driver file**.
6. Choose **XML Data File** in the **File Format** drop-down list.
7. Click **Finish**.
   The file appears in the Property Panel for you to define.
8. In the **Test Data Source** tab, specify the following in **File to use for data mapping:**
   `C:\210 Advanced Data Concepts\Data Files\XML Example1.xml`
9. In the **Production Data Source** tab, specify the following in **File to use in production:** **XMLdriver**
10. Save the file.

B. Map <id>

1. Drag the **XML Mapping** data file to the Edit Panel.
2. Double-click the <id> tag on the fourth line.
   Dialogue highlights the data in blue.
3. Right-click to access the shortcut menu and select **Tag** -> **Properties**.
   The **XML Tag Mapping Properties** dialog box opens.

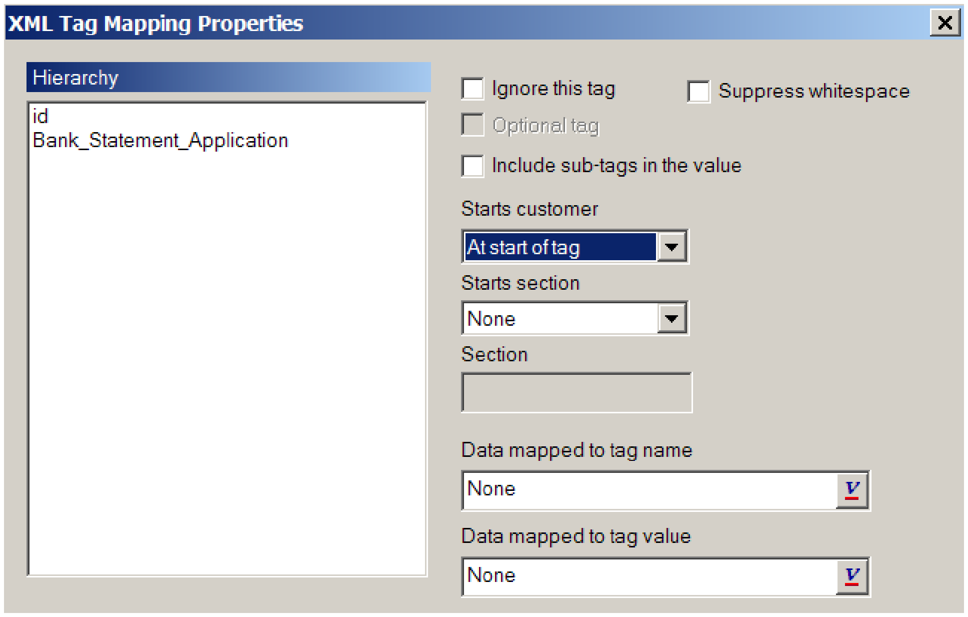
Choose Tag/Properties



```
<?xml version="1.0"?>
<!--Mapping for Bank Statement-->
<Bank_Statement_Application>
  <id
                 evens</CustomerName>
                       s">
                       0</SavingsStart>
                       </SavingsEnd>
             ate>9/15/03</DebitDate>
             ount>200.00</DebitAmount>
     <CreditDate>9/17/03</CreditDate>
     <CreditAmount>700.00</CreditAmount>
   </account>
   <account type="Checking">
```

4. At Starts customer, choose At start of tag.

XML Tag Mapping Properties (Top Portion)



id
Bank_Statement_Application

- [ ] Ignore this tag
- [ ] Suppress whitespace
- [ ] Optional tag
- [ ] Include sub-tags in the value

Starts customer: At start of tag

Starts section: None

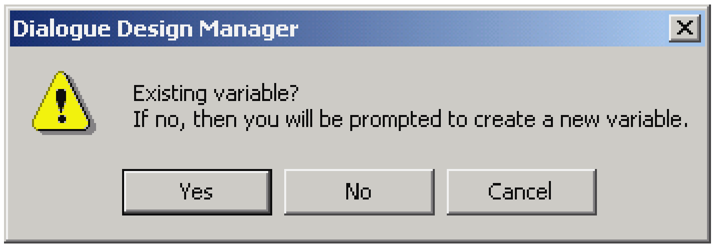
Section:

Data mapped to tag name: None

Data mapped to tag value: None

5. At **Data mapped to tag value**, click the variable button. The following dialog box appears.

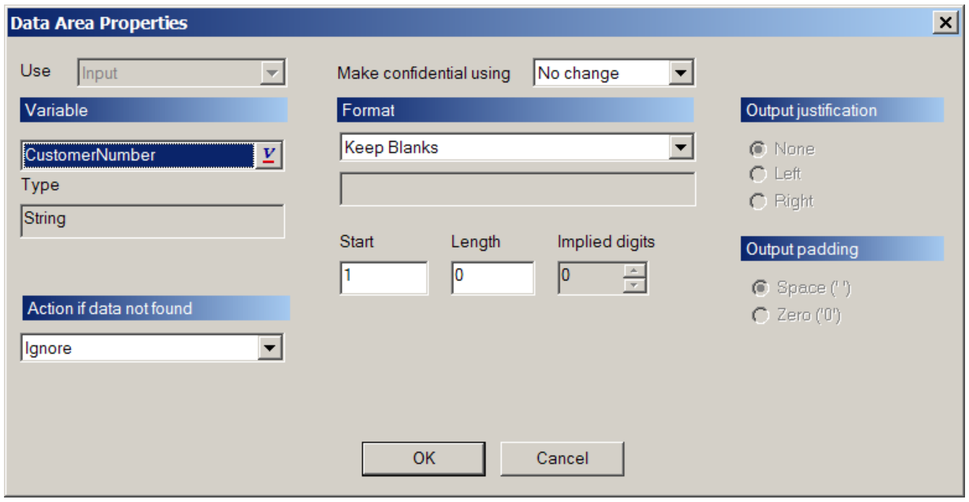
Message



Existing variable?
If no, then you will be prompted to create a new variable.

6. Click **Yes** to use a variable already defined.
   The **Select Variable** dialog box appears.
7. Choose CustomerNumber from the Exercise 14 XML Input folder.
8. Click OK to exit the dialog box.
   The Data Area Properties dialog box appears.

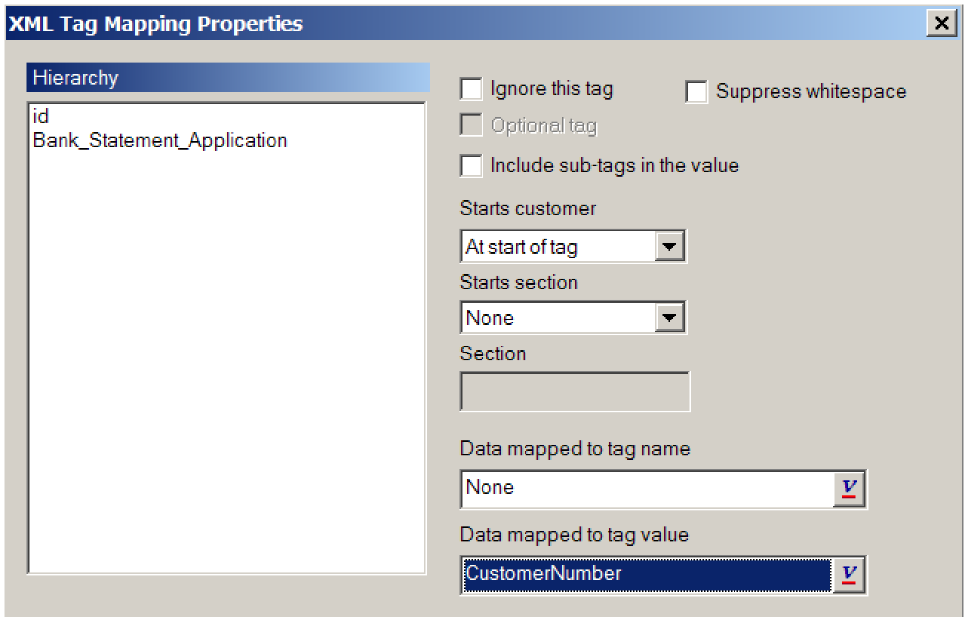
Data Area Properties Dialog Box



Use: Input
Variable: CustomerNumber
Type: String
Action if data not found: Ignore
Make confidential using: No change
Format: Keep Blanks
Start: 1
Length: 0
Implied digits: 0
Output justification: [✓ None] Left Right
Output padding: [✓ Space (' ')] Zero ('0')

9. The properties do not need changes, so click **OK**.
   The **XML Tag Mapping Properties** dialog box appears.

XML Properties Finished – Top Half Shown

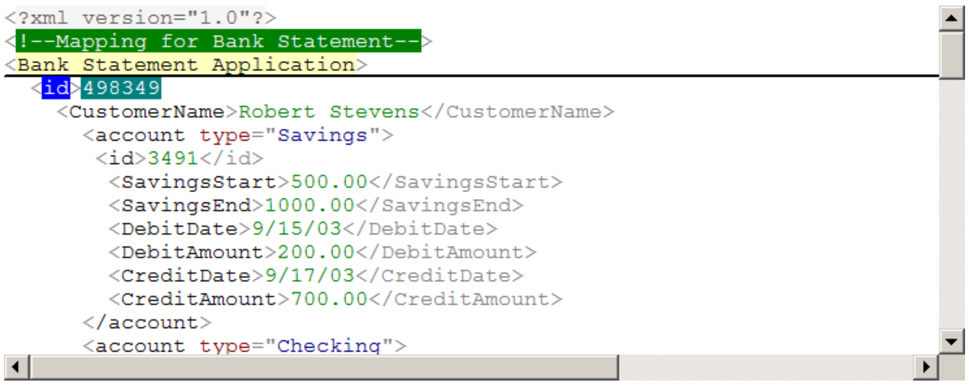


id
Bank_Statement_Application

Ignore this tag
Suppress whitespace
Optional tag
Include sub-tags in the value
Starts customer: At start of tag
Starts section: None
Section:
Data mapped to tag name: None
Data mapped to tag value: CustomerNumber

10. Click **OK** to finish mapping this tag.
    Dialogue highlights the tag value in teal.

Data File with first Tag Mapped to Tag Name

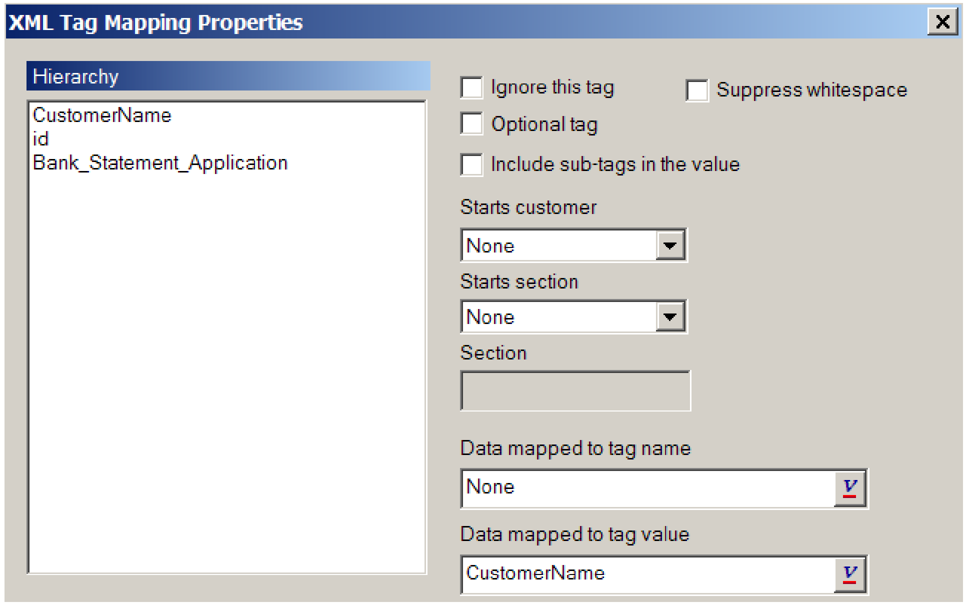


```
<?xml version="1.0"?>
<!--Mapping for Bank Statement-->
<Bank Statement Application>
  <id>498349
    <CustomerName>Robert Stevens</CustomerName>
      <account type="Savings">
       <id>3491</id>
         <SavingsStart>500.00</SavingsStart>
         <SavingsEnd>1000.00</SavingsEnd>
         <DebitDate>9/15/03</DebitDate>
         <DebitAmount>200.00</DebitAmount>
         <CreditDate>9/17/03</CreditDate>
         <CreditAmount>700.00</CreditAmount>
      </account>
      <account type="Checking">
```

C. Map the <CustomerName> Tag

1. Double-click the <CustomerName> tag.
   Dialogue highlights the data in blue.
2. Right-click to access the shortcut menu and select **Tag** -> **Properties**. The **XML Tag Mapping Properties** dialog box opens.
3. Clear the **Optional tag** check box.
4. In **Data mapped to tag value** choose **CustomerName** from the **Exercise 14 XML Input** folder as the variable.

XML Tag Mapping Properties Dialog Box



XML Tag Mapping Properties

Hierarchy
CustomerName
id
Bank_Statement_Application

[ ] Ignore this tag
[ ] Suppress whitespace
[ ] Optional tag
[ ] Include sub-tags in the value

Starts customer: None
Starts section: None
Section:
Data mapped to tag name: None
Data mapped to tag value: CustomerName

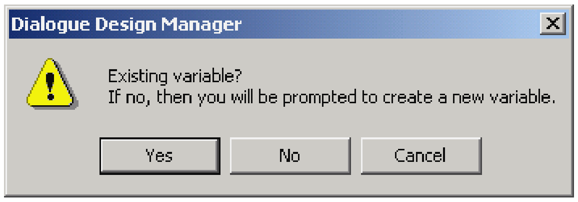
5. Click **OK** to exit the dialog box.

D. Map <account> to Start a Section

The next tag, <account>, starts a section.

1. Double-click the <account> tag.
   Dialogue highlights the data in blue.
2. Right-click to access the shortcut menu and select **Tag** -> **Properties**.
   The **XML Tag Mapping Properties** dialog box opens.
3. In the **Starts section** drop-down list select **Always**.
4. At **Section**, type `Savings`.
5. To map the attribute in this tag, do one of the following:
   j. If some attribute information appears in the **Attribute/Value Pairs** area, double-click the word **type** in the first column.
   k. If the attribute information does not appear, click the ✚ button.
   The XML Tag Attribute Properties dialog box appears.
6. Verify the word `type` appears for the **Attribute**.
7. Verify the word `Savings` appears for the **Value**.
8. Select the **The attribute/value pair is required for the mapping** check box.
9. At **Data mapped to value**, click the variable V button.
   A message appears.

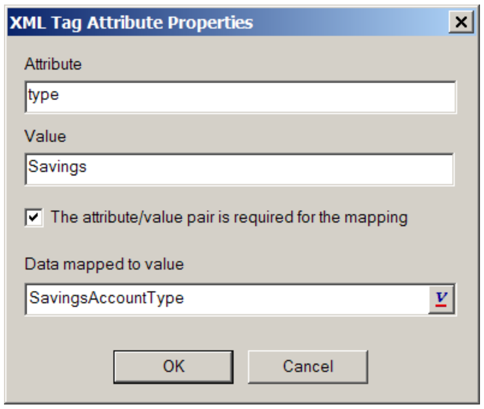
Message



Existing variable?
If no, then you will be prompted to create a new variable.

10. Click **Yes**.
    The **Select a Variable** dialog box appears.
11. Choose the **SavingsAccountType** variable from the **Exercise 14 XML Input** folder.
12. Click **OK** to return to **XML Tag Attributes Properties**.

XML Tag Attributes Properties



XML Tag Attribute Properties

Attribute: type

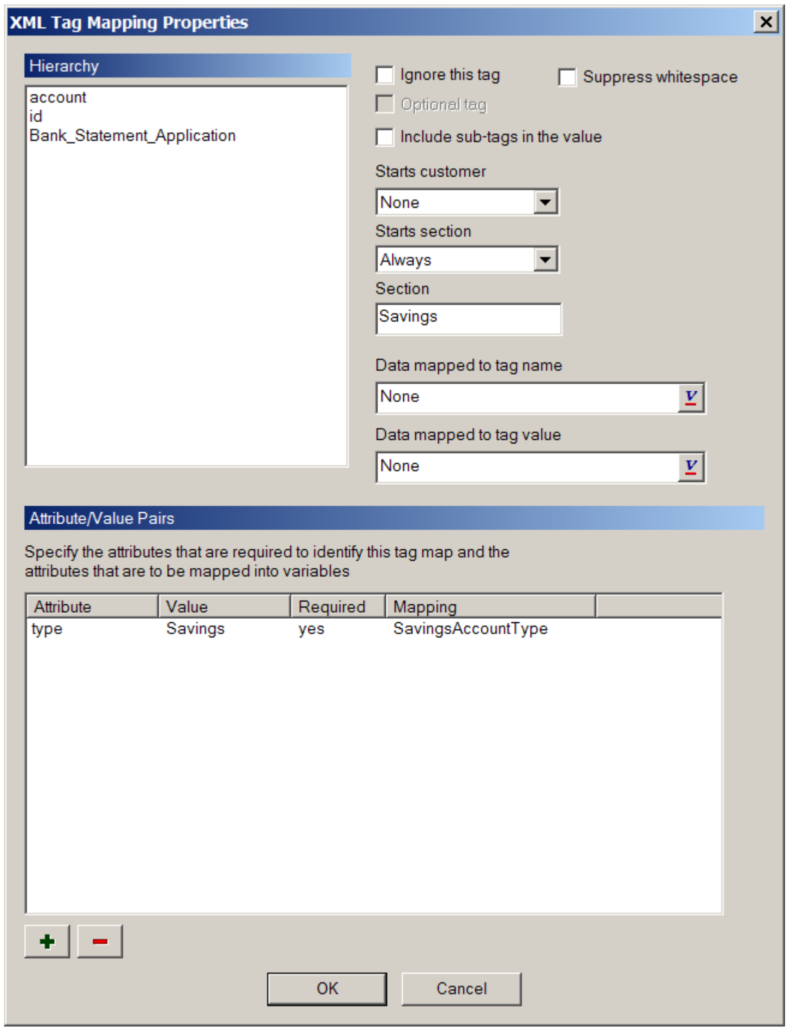
Value: Savings

☑ The attribute/value pair is required for the mapping

Data mapped to value: SavingsAccountType

13. Click **OK**.
    The **Data Area Properties** dialog box opes.
14. Click **OK** to return to the **XML Tag Mapping Properties**.

XML Tag Mapping Properties Dialog Box



XML Tag Mapping Properties

Hierarchy
account
id
Bank_Statement_Application

☐ Ignore this tag ☐ Suppress whitespace
☐ Optional tag
☐ Include sub-tags in the value

Starts customer: None

Starts section: Always

Section: Savings

Data mapped to tag name: None

Data mapped to tag value: None

Attribute/Value Pairs

Specify the attributes that are required to identify this tag map and the attributes that are to be mapped into variables

Attribute	Value	Required	Mapping
type	Savings	yes	SavingsAccountType

15. Click OK to close XML Tag Mapping Properties.

Dialogue visually shows the beginning of a section by creating a red line at this point in the Edit Panel.

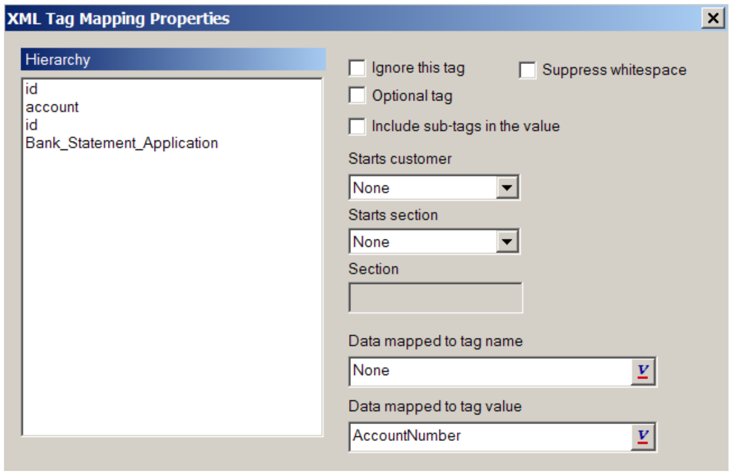
E. Map Next Three Tags

Map <id>

Map the <id> tag on the seventh line.

1. Clear the **Optional tag** check box.
2. Assign the **AccountNumber** variable from the **Exercise 14 XML Input** folder to this tag.

AccountNumber


XML Tag Mapping Properties

Hierarchy
id
account
id
Bank_Statement_Application

☐ Ignore this tag ☐ Suppress whitespace
☐ Optional tag
☐ Include sub-tags in the value
Starts customer: None
Starts section: None
Section:
Data mapped to tag name: None
Data mapped to tag value: AccountNumber

3. Leave the other properties at their default values and click **OK**.

Notice in the Edit Panel you have mapped the <id> tag in both the Savings and Checking sections.

Map <SavingsStart>

Map the <SavingsStart> tag on the eighth line.

4. Clear the **Optional tag** check box.
5. Assign the **SavingsBegin** variable from the **Exercise 14 XML Input** folder to this tag.
6. Leave the other properties at their default values and click **OK**.

Map <SavingsEnd>

Map the <SavingsEnd> tag on the ninth line.

7. Clear the **Optional tag** check box.
8. Assign the **SavingsEndingBalance** variable from the **Exercise 14 XML Input** folder to this tag.
9. Leave the other properties at their default values and click **OK**.

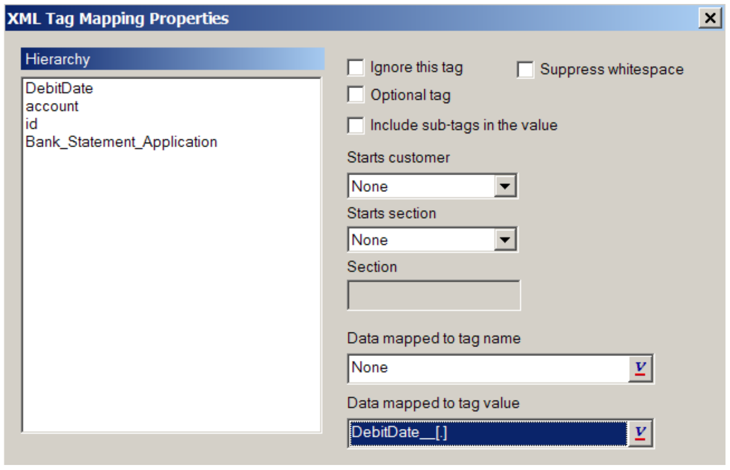
F. Map Duplicated Tags

The following tags appear in both the Savings and Checking sections.

Map <DebitDate>

1. Map the <DebitDate> tag on the tenth line.
2. In the XML Tag Mapping Properties dialog box, clear the Optional tag check box.
3. Assign the DebitDate_ array variable from the Exercise 14 XML Input folder to this tag.
4. In the Data Area Properties dialog box, specify m/d/yy (4/6/01) as the Format. This matches the date format in the Customer driver file.
5. In the Data Area Properties dialog box, click OK to return to XML Tag Mapping Properties.

XML Tag Mapping Properties Dialog Box



Hierarchy: DebitDate, account, id, Bank_Statement_Application

Starts customer: None
Starts section: None
Data mapped to tag name: None
Data mapped to tag value: DebitDate__[]

6. In the XML Tag Mapping Properties dialog box, click OK.

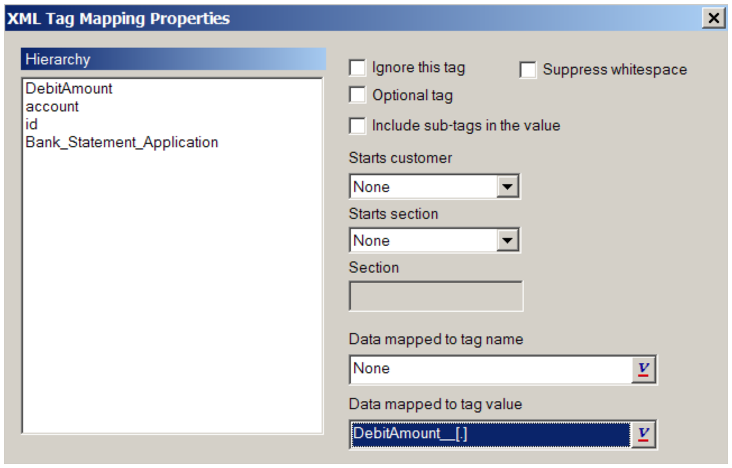
In the Edit Panel note how you have mapped the <DebitDate> in both the Savings and Checking sections.

Map <DebitAmount>

Map the <DebitAmount> tag on the eleventh line.

1. In XML Tag Mapping Properties, clear the Optional tag check box.
2. Assign the **DebitAmount_** array variable from the **Exercise 14 XML Input** folder to this tag.

XML Tag Mapping Properties


XML Tag Mapping Properties

Hierarchy
DebitAmount
account
id
Bank_Statement_Application

☐ Ignore this tag ☐ Suppress whitespace
☐ Optional tag
☐ Include sub-tags in the value

Starts customer: None
Starts section: None
Section:
Data mapped to tag name: None
Data mapped to tag value: DebitAmount__[]

3. Click **OK**.

In the Edit Panel note how you have mapped the <DebitAmount> in both the Savings and Checking sections.

Map <CreditDate>

Map the <CreditDate> tag on the twelfth line.

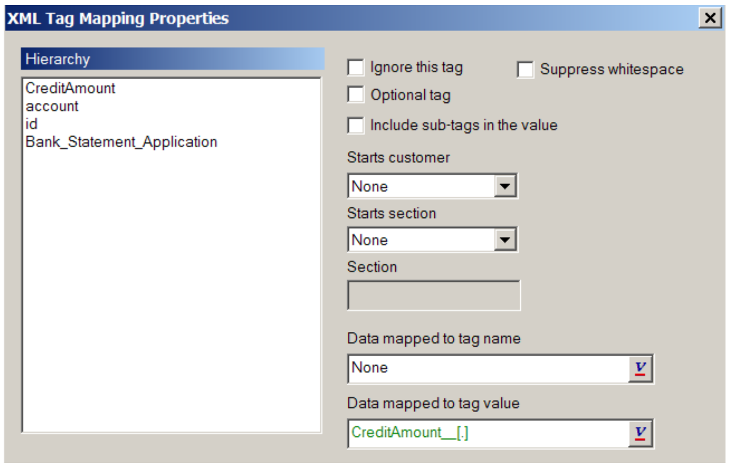
1. Clear the **Optional tag** check box.
2. Assign the **CreditDate_** array variable from the **Exercise 14 XML Input** folder to this tag.
3. Click **OK**.
4. In the **Data Area Properties** dialog box, specify **m/d/yy (4/6/01)** as the **Format**.
5. Click **OK**.
6. In **XML Tag Mapping Properties**, click **OK**.

Map <CreditAmount>

Map the <CreditAmount> tag on the thirteenth line.

7. Clear the **Optional tag** check box.
8. Assign the **CreditAmount_** array variable from the **Exercise 14 XML Input** folder to this tag.

XML Tag Mapping Properties



CreditAmount
account
id
Bank_Statement_Application

☐ Ignore this tag
☐ Suppress whitespace
☐ Optional tag
☐ Include sub-tags in the value

Starts customer
None
Starts section
None
Section
Data mapped to tag name
None
Data mapped to tag value
CreditAmount__[.]

9. Click **OK**.
10. **Save** your mapping so far. Do not exit the Edit Panel.

G. Map <Checking> Section

Second <account> tag

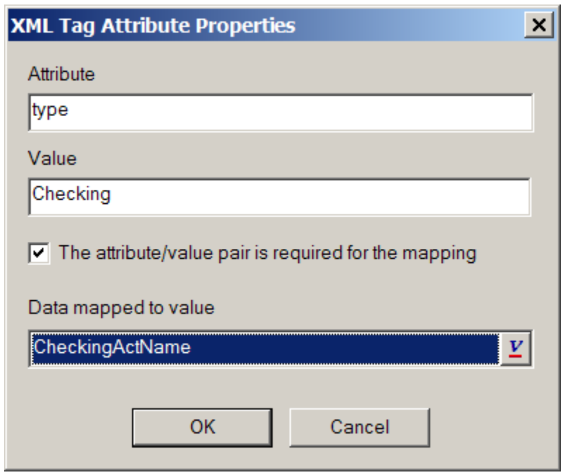
The second **<account>** *start tag* in the Edit Panel begins a separate section.

1. Highlight the unmapped **<account>** tag (the one with the type="Checking" attribute) and progress to the **XML Tag Mapping Properties** dialog box.
2. In the **Start section** drop-down list select **Always**.
3. At **Section**, type `Checking`.
4. Highlight the word type in the attribute area, and click the − button to remove the savings attribute from this mapping.
5. To map the checking attribute in this tag, click the ✚ button.

The XML Tag Attribute Properties dialog box appears.

6. Enter the word `type` in the **Attribute** box.
7. Enter `Checking` in the **Value** box.
8. Select the **The attribute/value pair is required for the mapping** check box.
9. At **Data mapped to value**, map the **CheckingActName** variable from the **Exercise 14 XML Input** folder.
10. Click **OK**.
    The **Data Area Properties** dialog box opens.
11. Click **OK** to return to **XML Tag Attributes Properties**.

Attribute/Value Pairs Dialog Box


XML Tag Attribute Properties

Attribute: type

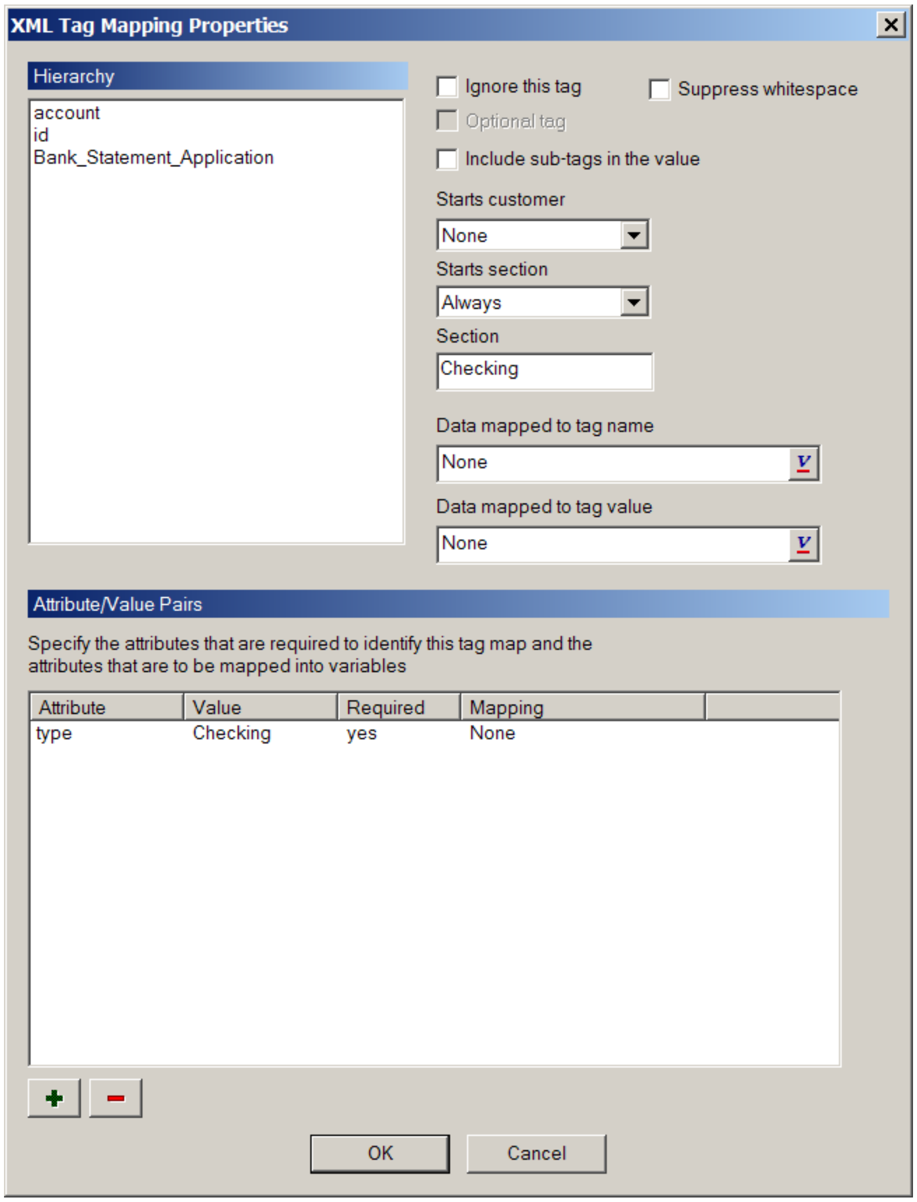
Value: Checking

☑ The attribute/value pair is required for the mapping

Data mapped to value: CheckingActName

12. Click OK to return to XML Tag Mapping Properties.

XML Tag Mapping Properties Dialog Box



XML Tag Mapping Properties

Hierarchy

account
id
Bank_Statement_Application

- [ ] Ignore this tag
- [ ] Suppress whitespace
- [ ] Optional tag
- [ ] Include sub-tags in the value

Starts customer: None

Starts section: Always

Section: Checking

Data mapped to tag name: None

Data mapped to tag value: None

Attribute/Value Pairs

Specify the attributes that are required to identify this tag map and the attributes that are to be mapped into variables

Attribute	Value	Required	Mapping
type	Checking	yes	None

13. Click OK to exit XML Tag Mapping Properties.

Since this begins a section, Dialogue places a red line at this point in the Edit Panel.

Map <CheckingStart>

Map the <CheckingStart> tag in the Checking section.

1. Clear the **Optional tag** check box.
2. Assign the **CheckingStart** variable from the **Exercise 14 XML Input** folder to this tag.

3. Click **OK**.

Map <CheckingEnd>

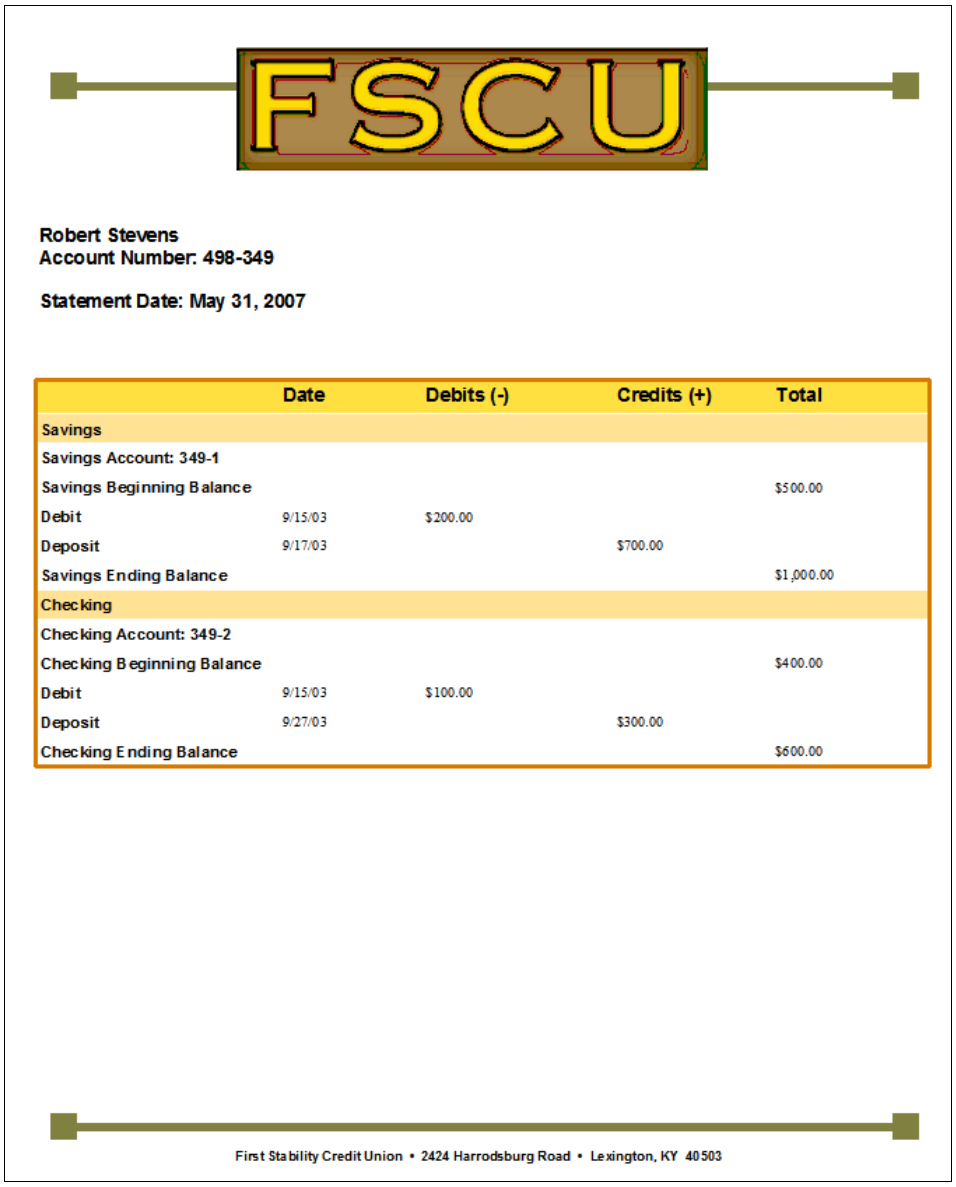
Map the <CheckingEnd> tag in the Checking section.

4. Clear the **Optional tag** check box.
5. Assign the **CheckingEndBalance** variable from the **Exercise 14 XML Input** folder to this tag.
6. Click **OK**.
7. You have completed mapping the tags in this data file. Save and close the Edit Panel.

H. Compose Output

1. Under the **Exercise 14 XML Input** folder, expand the **Applications** heading and the **Data Files** heading.
2. Drag the **XML Mapping** data file to the **XML Mapping Application**.
3. Package the application and view the results in **Designer**.

Composed Output



FSCU

Robert Stevens
Account Number: 498-349

Statement Date: May 31, 2007

	Date	Debits (-)	Credits (+)	Total
Savings				
Savings Account: 349-1				
Savings Beginning Balance				$500.00
Debit	9/15/03	$200.00		
Deposit	9/17/03		$700.00	
Savings Ending Balance				$1,000.00
Checking				
Checking Account: 349-2				
Checking Beginning Balance				$400.00
Debit	9/15/03	$100.00		
Deposit	9/27/03		$300.00	
Checking Ending Balance				$600.00

First Stability Credit Union • 2424 Harrodsburg Road • Lexington, KY 40503

Chapter 15: Dynamic Data Access

15.1 In this Chapter

With Dynamic Data Access (DDA) and data connectors, the Dialogue Engine can be linked directly to third party message queues and enterprise systems for real-time data access. Data from customer service systems, web applications and other types of systems can be integrated into the document creation process, resulting in real-time, highly personalized communications delivered through multiple channels.

In this chapter you will learn to connect to any corporate database using the Dynamic Data Access module.

Objectives

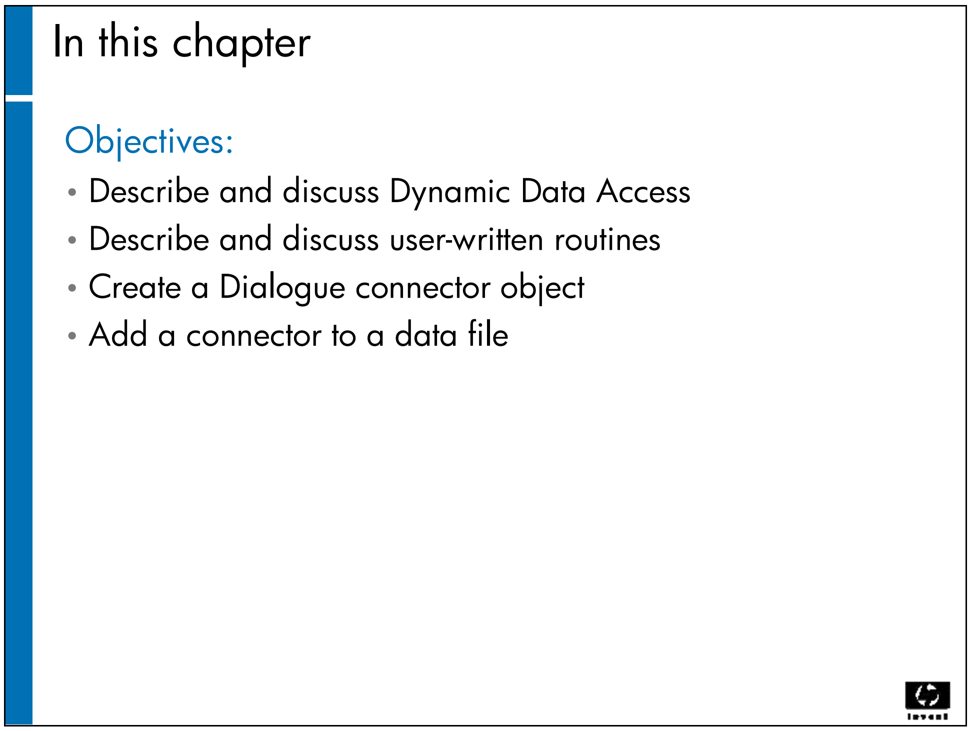
By the end of this chapter you will:

- Describe and discuss Dynamic Data Access
- Describe and discuss user-written functions.
- Create a Dialogue Connector object.
- Add a Connector to a Data File.

For more information

For more information, refer to the following Dialogue documentation.

- Dynamic Data Access guide DDA.pdf



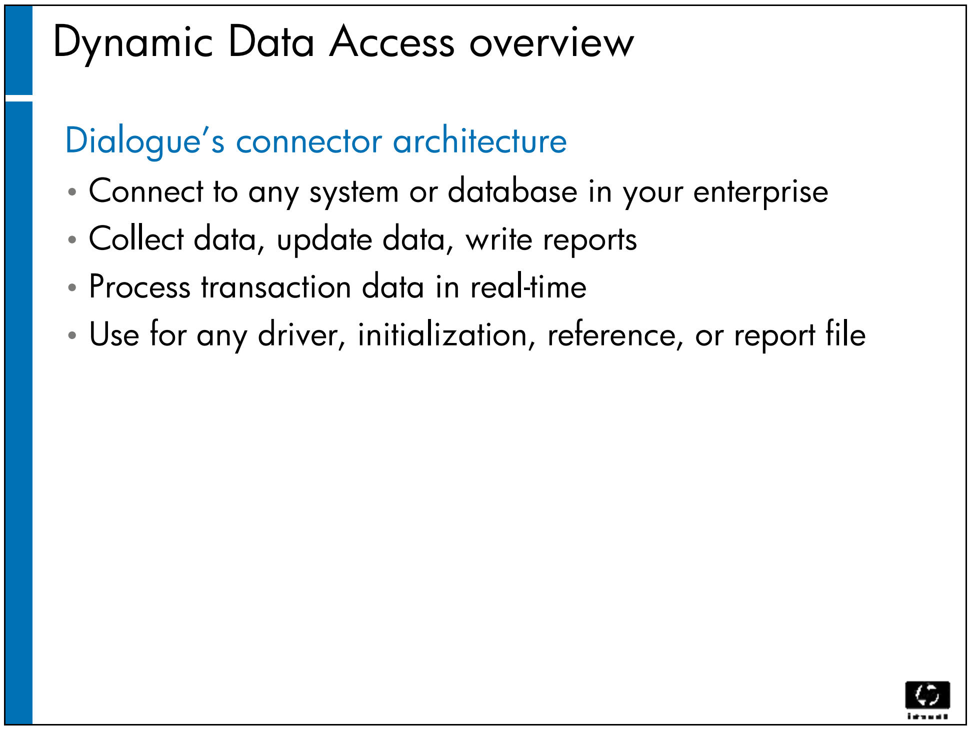
In this chapter

Objectives:

- Describe and discuss Dynamic Data Access
- Describe and discuss user-written routines
- Create a Dialogue connector object
- Add a connector to a data file

15.2 Dynamic Data Access Overview

Dynamic Data Access



Dynamic Data Access overview

Dialogue's connector architecture

- Connect to any system or database in your enterprise
- Collect data, update data, write reports
- Process transaction data in real-time
- Use for any driver, initialization, reference, or report file

Dynamic Data Access (DDA) is Dialogue's connector architecture module. It connects Dialogue's Engine to any system or database in your enterprise infrastructure to collect data, update data, write reports, or execute user-written routines. This allows Dialogue to process transaction data in real-time, support encryption/decryption applications, and read/write to any corporate database or application. DDA can be used for any or all of the Dialogue driver, initialization, reference, and report files. DDA allows table lookup on the fly and can spawn other processes.

Features

- An application programming interface (API) to retrieve data (input files or dynamic images and text), write report files, and write to output queues.
- Allows users to write their own connectors to retrieve data (input files or dynamic images and text), write report files, and write to output queues, or purchase existing connectors built on the DDA architecture.
- Language support for:
  - Mainframe: C, C++, COBOL, Fortran, PL1, Assembler
  - Windows: any language that can create a DLL
  - UNIX and AS/400: any language that create a shared object (.so)

- Allows Dialogue's Engine to operate in a transaction server mode where DDA waits for the transaction data.

Process overview

- Write a function that does the required processing
- Create a Dialogue connector object.
- Assign the connector to a file, placeholder, output, or queue object.

15.3 Writing Connector Routines

Parameters

All Connector routines must accept the following four arguments.

- **Address of Maximum Buffer Size**
  unsigned long * ulMaxBufferSize
  This size must be greater than the number of bytes in the largest record in the file.
- **Address of Current Buffer Size (*or* seek address)**
  unsigned long * ulBufferSize
- **Address of Buffer**
  char * szBuffer
- **Address of Access Type**
  unsigned long * ulAccessType
  - Can contain one of the following integer values

    0 - Read file

    1 - Write to file

    2 - Update data file

    3 - Seek on key value

    4 - Open file

    szBuffer parameters contain a string with the Dialogue file name

    5 - Close file

    6 - Clear file

    7 - End of transaction

    sent to report files after data file returns End of File

    8 - End of customer

    sent to report files after Engine processes each customer

A fifth parameter, **Address of User-Defined Pointer**, is optional. You set this value in the Open function. The Engine passes this value to memory so the DDA function can refer back to its opening parameters.

CR/LF

With *input*, Dialogue automatically places CR/LF on DDA records in Windows and Unix environments (with the record-based MVS platform, no ending values are required). For this reason, your Connector routine should *not* place CR/LF on records.

With *output*, Dialogue writes to the record buffer twice: the first write supplies the data and the second write supplies a CR/LF.

Returned Values

The export function will return one of the following values to the Engine.

0 - Data returned (success)

1 - Buffer overflow - bytes read greater than ulMaxBufferSize (error)

2 - End of file (EOF)

3 - Unable to open file

4 - Shut down Engine (transaction mode only)

5 - Flush report files (transaction mode only)

6 - EOF "but keep Engine going" – Transaction mode

7 - Files closed

Access you must Support in your Routines

	Initialization File	**Customer Driver File**	**Reference File**	**Report File**	**Placeholder Variable**
Open	Yes	Yes	Yes	Yes	Yes
Close	Yes	Yes	Yes	Yes	Yes
Seek			Yes		Yes
Read	Yes	Yes	Yes		Yes
Write				Yes	
Update		Future	Future		
End of Customer				Optional	
End of Transaction				Optional	

Requirements for Windows DLL

The dll must have an export entry for each exported function. For example, if the dll uses the function getRec, the .def file must include an export entry, such as:

```
GetRec @1
```

The following summarizes Connector routine programming considerations:

When the Engine runs in continuous transaction mode, a processing event causes data to be returned to a record buffer. The programmer's task is to write a script so the Engine will process this data.

- The dll script must perform blocking.
- It can return an "end of customer" indicator instead of the normal first record of the next customer.
- To create one output file per customer, use output queues set to break for each customer.
- The script can also return an "interim message files" indicator so the Engine produces multiple status files (message and report files).

Export Function

First, determine what data file **Type** the dll most resembles. This choice determines whether you use Export Function getRec or writeRec.

DLL Type	**Export Function**
Initialization File	getRec
Customer Driver File	
Reference File	
Placeholder Variable	
Report File	writeRec
Output File	

```
getRec(unsigned long*ulMaxBufferSize,unsigned long*ulBufferSize,
char*szBuffer,unsigned long*ulAccessType)
```

```
writeRec(unsigned long*ulMaxBufferSize,unsigned long*ulBufferSize, char *
szBuffer,unsigned long *ulAccessType)
```

A fifth parameter, Address of User-Defined Pointer, is optional.

15.4 Creating Connector Objects

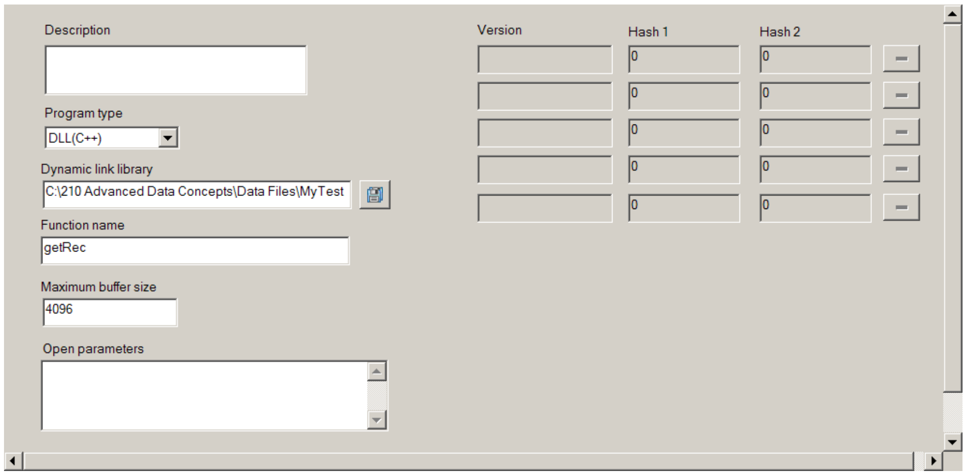
Connector objects in the Library provide the Engine information about the location, name, and operating parameters of an external routine. These objects appear in *Connectors* headings in the Design Manager Library. Create them by right-clicking the Library heading, and selecting **New Connector** from the shortcut list..

Connector Properties

To complete the properties for this object, you need the following information from the programmer who wrote the associated Connector routine:

- Program type
  - For all platforms, except MVS, choose **DLL(C++)**
  - For MVS, choose: **C**, **COBOL**, **FORTRAN**, **ASM (Assembler)**, **PL1**
- Dynamic link library
  Identify the location and name of your user routine. On a windows platform you can use the button to browse and select the Connector.
- **Function name**
  Type the name of the user-written **Function Name**. For COBOL only, the name of the Dynamic Link Library and Function Name must be the same.
- **Maximum buffer size**
  You type here the largest buffer size permitted. This value must be greater than the largest record in the data file. Your routine can read any number of bytes, as long as a single call returns at least one record. By specifying a record buffer that is greater than the largest record, you reduce processing time.
- **Open parameters**
  What you type here passes to the Connector routine with the Open access parameter. One example of its use: you can identify different initialization files (INIFILEs) here so the WebSphere MQ Connector can access multiple Output Queues.

Connector Properties



Description

Program type
DLL(C++)

Dynamic link library
C:\210 Advanced Data Concepts\Data Files\MyTest

Function name
getRec

Maximum buffer size
4096

Open parameters

Version	Hash 1	Hash 2
	0	0
	0	0
	0	0
	0	0
	0	0

15.5 Adding Connectors to Data Files

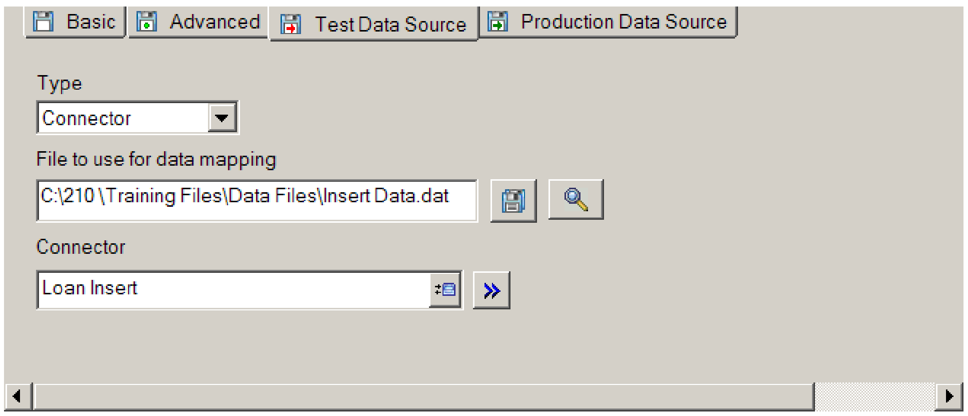
You define a customer driver, reference, initialization, or other data file for DDA as you do other data file objects in the Library. The **Basic** tab and the **Advanced** properties need not change. The differences occur on the **Data Source** tabs.

Data Source Tabs

The major difference between a DDA data file and a standard data file is your selection of **Connector** for a data source **Type** on the **Data Source** tabs of the data file properties. When you select **Connector**, the **Connector** property appears.

From the **Connector** dropdown list, select the Library Connector object for your Data File.

Data Source Tab Properties



Type
Connector

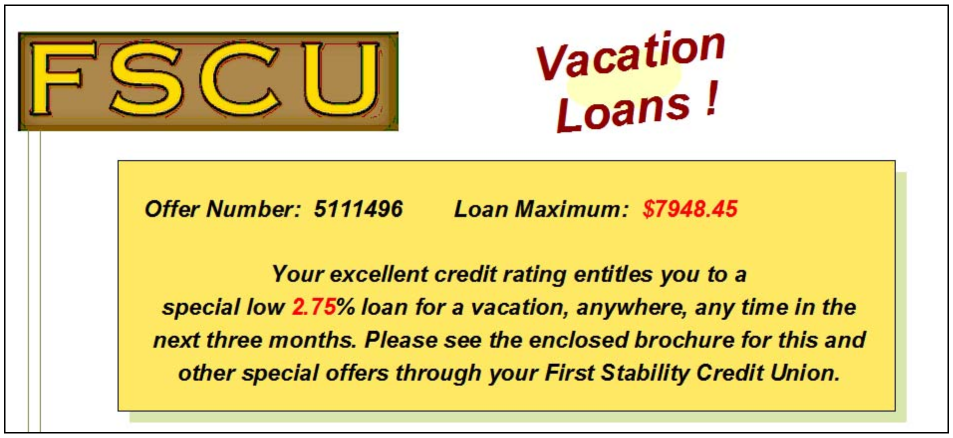
File to use for data mapping
C:\210\Training Files\Data Files\Insert Data.dat

Connector
Loan Insert

15.6 Exercise: Use Dynamic Data Access

This exercise demonstrates using dynamic data access to create output. You will produce a personalized bill insert using dynamic data access for the customer data.

DDA Composed Output


FSCU

Vacation Loans !

Offer Number: 5111496 *Loan Maximum: $7948.45*

Your excellent credit rating entitles you to a special low 2.75% loan for a vacation, anywhere, any time in the next three months. Please see the enclosed brochure for this and other special offers through your First Stability Credit Union.

A. Preview the Application

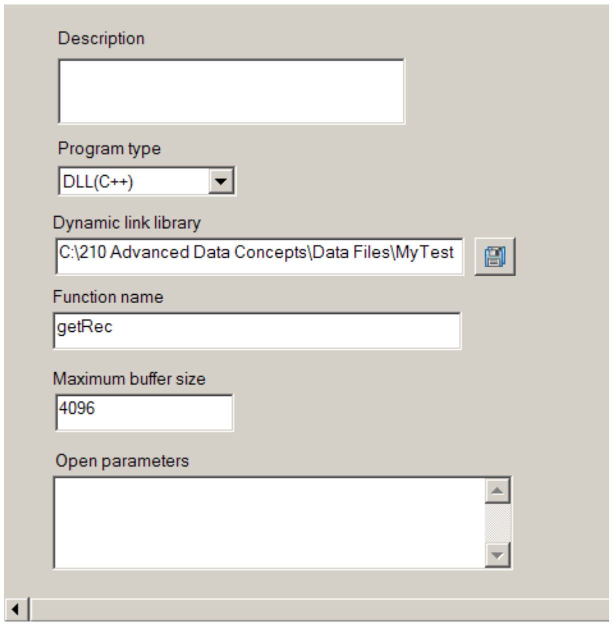
1. In the Library, expand the **Exercise 15 Dynamic data access** folder and its **Applications** heading.
2. Expand the **Vacation Loan Insert** application, and drag the **Dynamic Data Access** data file to the Property Panel.
3. Note the **Test** and **Production Data Source** files.
4. Package the application and view the results in the Viewer.

B. Create the Connector

1. Right-click the **Connectors** heading and select **New Connector** from the shortcut menu.
   The **New Connector** dialog box appears.
2. Type `Loan Insert` in the **Name** property and click **Finish**.
   The new connector saves and appears in the property panel for you to complete.
3. Verify that **Program type** displays the default of `DLL(C++)`.
4. At **Dynamic link library**, specify the following:
   ```
   C:\210 Advanced Data Concepts\Data
   Files\MyTestDLL.dll
   ```
5. Type `getRec` as **Function name**.
6. Type `4096` as **Maximum buffer size**.

7. Save and close the Property Panel.

Loan Insert (Connector)


Description

Program type
DLL(C++)

Dynamic link library
C:\210 Advanced Data Concepts\Data Files\MyTest

Function name
getRec

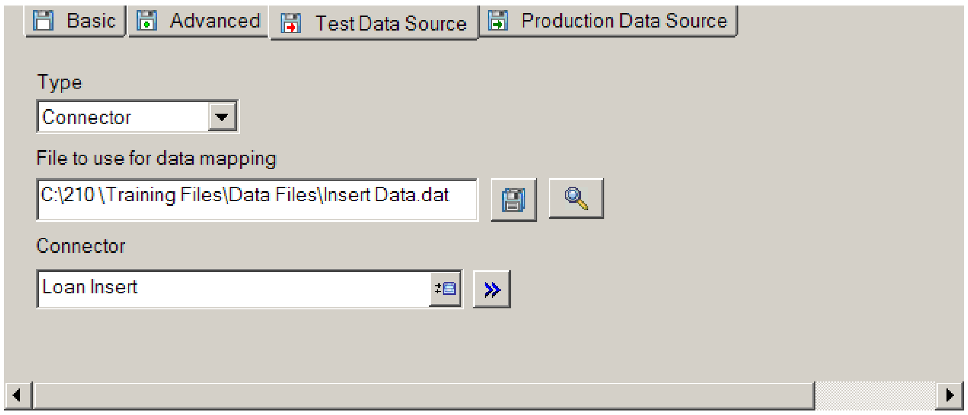
Maximum buffer size
4096

Open parameters

C. Update the Data File

1. Drag the **Dynamic Data Access** data file to the Property Panel.
2. On the **Production Data Source** tab, change **Type** to `Connector`.
3. For **Connector**, select the `Loan Insert` connector that you just created.
4. Save and close the Property Panel.

Data Source Tab



Type
Connector

File to use for data mapping
C:\210\Training Files\Data Files\Insert Data.dat

Connector
Loan Insert

D. Test the Result

1. Package and run the **Vacation Loan Insert** application.

2. Review the results in the Viewer.

Trademarks