

OpenText™ Exstream™

Using Data to Drive an Application

Design and Production Documentation

Release 16.6.0

OpenText™ Exstream
Using Data to Drive an Application
Rev.: 2019-Apr-30

This documentation has been created for software version 16.6.0.

It is also valid for subsequent software versions as long as no new document version is shipped with the product or is published at <https://knowledge.opentext.com>.

Open Text Corporation
275 Frank Tompa Drive, Waterloo, Ontario, Canada, N2L 0A1

Tel: +1-519-888-7111
Toll Free Canada/USA: 1-800-499-6544 International: +800-4996-5440
Fax: +1-519-888-0677

Support: <https://support.opentext.com>
For more information, visit <https://www.opentext.com>

Copyright © 2019 Open Text. All rights reserved.

Trademarks owned by Open Text.

One or more patents may cover this product. For more information, please visit, <https://www.opentext.com/patents>

Disclaimer

No Warranties and Limitation of Liability

Every effort has been made to ensure the accuracy of the features and techniques presented in this publication.
However, Open Text Corporation and its affiliates accept no responsibility and offer no warranty whether expressed or implied, for the accuracy of this publication.

Contents

| | |
|--|----|
| Chapter 1: Using Data in Exstream Design and Production | 1 |
| 1.1 Data in Exstream Design and Production | 2 |
| 1.2 Data-Related Modules | 3 |
| 1.3 Section Data in Exstream Design and Production | 4 |
| 1.4 XML Nodes in Exstream Design and Production | 6 |
| Chapter 2: Setting Up Data Files | 9 |
| 2.1 Data File Formats and Types | 10 |
| 2.1.1 Data File Formats | 10 |
| 2.1.2 Data File Types | 12 |
| 2.1.3 Data File Type and Format Availability | 14 |
| 2.2 Parts of a Data File | 16 |
| 2.2.1 Record Type Indicators | 17 |
| 2.3 Creating an Auxiliary Layout Data File | 18 |
| 2.3.1 Creating an Auxiliary Layout File Object | 19 |
| 2.3.2 Defining the Auxiliary Layout Format | 20 |
| 2.4 Creating a Customer Driver Data File | 21 |
| 2.4.1 Creating a Customer Driver File Object | 22 |
| 2.4.2 Defining the Customer Driver File Format | 23 |
| 2.4.3 Defining the Records in a Customer Driver File | 25 |
| 2.4.4 Selecting the Customer Driver File Data Source | 27 |
| 2.5 Creating an Initialization or a Post-sort Initialization Data File | 29 |
| 2.5.1 Creating an Initialization or a Post-sort Initialization File Object | 30 |
| 2.5.2 Defining the Initialization or Post-sort Initialization File Format | 31 |
| 2.5.3 Defining the Records in an Initialization or a Post-sort Initialization File | 33 |
| 2.5.4 Selecting the Initialization or Post-sort Initialization File Data Source | 35 |
| 2.6 Creating a Reference Data File | 37 |
| 2.6.1 Creating a Reference File Object | 38 |
| 2.6.2 Defining the Reference File Format | 39 |
| 2.6.3 Defining the Records in a Reference File | 45 |
| 2.6.4 Selecting the Reference File Data Source | 47 |
| 2.7 Creating a Report or a Post-sort Report File | 50 |
| 2.7.1 Creating a Report or Post-sort Report File Object | 51 |

| | |
|--|-----------|
| 2.7.2 Defining the Report or Post-sort Report File Format | 52 |
| 2.7.3 Defining the Records in a Report or Post-sort Report File | 55 |
| 2.7.4 Selecting the Report or Post-sort Report File Template and Result Location | 56 |
| Chapter 3: Setting Up Variables | 59 |
| 3.1 Types of Variables | 60 |
| 3.1.1 Variable Sources | 60 |
| 3.1.2 Data Type Represented by a Variable | 62 |
| 3.1.3 Number of Values a Variable Can Represent | 63 |
| 3.2 Creating Variables | 64 |
| 3.2.1 Creating a Variable Object | 64 |
| 3.2.2 Specifying the Variable Values | 65 |
| 3.2.3 Selecting Variable Validation and Troubleshooting Methods | 72 |
| 3.2.4 Tagging Variables with Metadata | 74 |
| 3.2.5 Creating Variables Automatically from XML or an XML Schema | 74 |
| 3.2.6 Populating a String Variable Using a String Lookup Table | 78 |
| 3.2.7 Adding Variables to an Application | 79 |
| 3.2.8 Specifying How Array Variable Elements are Displayed | 80 |
| 3.3 Formatting Variable Values in Output | 81 |
| 3.3.1 Formatting Boolean Values | 82 |
| 3.3.2 Formatting Currency Values | 83 |
| 3.3.3 Formatting Date Values | 86 |
| 3.3.4 Formatting Floating Values | 87 |
| 3.3.5 Formatting Integer Values | 91 |
| 3.3.6 Formatting String Values | 94 |
| 3.3.7 Formatting Tagged Text Values | 96 |
| 3.3.8 Creating a Custom Format String | 97 |
| 3.4 Testing Variables | 99 |
| 3.4.1 Testing a Variable to Ensure Correct Data Population | 99 |
| 3.4.2 Creating a Report to List the Variables Used in an Application | 101 |
| 3.5 System Variables | 103 |
| 3.5.1 Application Design Related System Variables | 104 |
| 3.5.2 Counting and Numbering Related System Variables | 106 |
| 3.5.3 Customer Related System Variables | 113 |
| 3.5.4 Data Related System Variables | 115 |
| 3.5.5 Date Related System Variables | 117 |
| 3.5.6 Design Related System Variables | 118 |
| 3.5.7 High-Volume Output Processing Related System Variables | 119 |

| | |
|--|------------|
| 3.5.8 Marketing Related System Variables | 124 |
| 3.5.9 Production Related System Variables | 126 |
| 3.5.10 Real-time and SOAP-Related System Variables | 127 |
| Chapter 4: Data File Mapping | 129 |
| 4.1 Setting Up the Data Mapping View | 130 |
| 4.1.1 Selecting the Default Data Mapping View of the Property Panel | 130 |
| 4.1.2 Limiting the Data Records That Appear in the Edit Panel | 133 |
| 4.1.3 Changing the Text Encoding in the Edit Panel | 133 |
| 4.1.4 Changing the Font in the Edit Panel | 134 |
| 4.1.5 Viewing the Hexadecimal Value of the Text | 134 |
| 4.2 Automapping a Data File | 135 |
| 4.2.1 Automapping a Columnar Data File | 135 |
| 4.2.2 Automapping a Delimited Data File | 140 |
| 4.3 Manually Mapping a Data File | 146 |
| 4.3.1 Mapping a Variable to Sample Data in a Data Area | 146 |
| 4.3.2 Mapping a Data Area with No Sample Data in a Delimited Data File | 147 |
| 4.3.3 Mapping Multiple Variables to a Single Data Area | 148 |
| 4.3.4 Mapping Groups of Data Areas That Repeat in a Single Record | 149 |
| 4.3.5 Copying the Data Map Layout of a Record | 150 |
| 4.4 Using a Data Area to Specify How the Engine Reads and Writes Data | 151 |
| 4.4.1 Starting a New Customer, a New Section, or a New Recipient | 151 |
| 4.4.2 Specifying How the Engine Uses a Data Area During Processing | 153 |
| 4.4.3 Specifying the Input Format of Data in a Data Area | 154 |
| 4.4.4 Specifying When the Engine Writes Data to a Report File | 156 |
| 4.5 Modifying a Data Map | 157 |
| 4.5.1 Resizing the Length of a Data Map in a Columnar Data File | 157 |
| 4.5.2 Moving a Data Map | 158 |
| 4.5.3 Mapping a New Variable to a Data Area | 158 |
| 4.5.4 Removing a Data Map | 159 |
| 4.6 Testing a Data Map | 159 |
| 4.7 Searching a Mapped Data File | 160 |
| 4.7.1 Finding Specific Text | 160 |
| 4.7.2 Finding a Specific Variable | 161 |
| 4.7.3 Finding the Start of a Customer | 162 |
| 4.7.4 Finding the Start of a Section | 162 |
| 4.8 Data Area Input Formats | 162 |
| 4.8.1 Boolean Data Area Input Formats | 163 |

| | |
|---|-----|
| 4.8.2 Currency, Floating, and Integer Data Area Input Formats | 164 |
| 4.8.3 Date Data Area Input Formats | 167 |
| 4.8.4 Formatted Text Data Area Input Formats | 167 |
| 4.8.5 Placeholder Data Area Input Formats | 168 |
| 4.8.6 String and Tagged Text Data Area Input Formats | 172 |
| 4.8.7 Creating Custom Data Area Input Formats | 176 |
| Chapter 5: Setting Up XML Formatted Data Files | 178 |
| 5.1 Module Requirements for XML Data Source Formats | 178 |
| 5.2 Creating XML Formatted Data Files | 179 |
| 5.2.1 Considerations for Setting Up an XML Formatted Data Source | 179 |
| 5.2.2 Creating a Data File Object | 184 |
| 5.2.3 Defining the Data Mapping Method | 185 |
| 5.2.4 Selecting the Data File Sources | 186 |
| 5.2.5 Mapping an XML Formatted Data File | 189 |
| 5.2.6 Validating a Data Source at Testing and Run Time Using a Schema | 202 |
| 5.2.7 Transforming Data at Testing and Run Time | 205 |
| 5.2.8 Using Attributes to Identify the XML Hierarchy | 208 |
| 5.2.9 Defining the Structure of an XML Formatted Output Data File | 209 |
| 5.2.10 Using a Variable Report to Create a Sample XML Data Layout | 210 |
| 5.2.11 Integrating XML Schemas with XML Formatted Data Files | 212 |
| 5.3 Creating Schema Model Data Files | 215 |
| 5.3.1 Considerations for Setting Up a Schema Model Data Source | 216 |
| 5.3.2 Creating a Schema Model Data File Object | 220 |
| 5.3.3 Mapping a Schema Model Data File | 222 |
| 5.3.4 Changing Schema Model Data File Sources | 230 |
| 5.3.5 Validating a Data Source at Testing and Run Time Using a Schema | 232 |
| 5.3.6 Transforming Data at Testing and Run Time | 235 |
| 5.3.7 Using XML Nodes to Drive an Application | 238 |
| Chapter 6: Setting Up JSON Formatted Data Files | 253 |
| 6.1 Module Requirements for JSON Data Source Formats | 253 |
| 6.2 Creating JSON Formatted Data Files | 253 |
| 6.2.1 Considerations for Setting Up a JSON Formatted Data Source | 254 |
| 6.2.2 Creating a Data File Object | 255 |
| 6.2.3 Defining the Data Mapping Method | 256 |
| 6.2.4 Selecting the Data File Sources | 257 |
| 6.2.5 Mapping a JSON Formatted Data File | 260 |

| | |
|--|------------|
| 6.2.6 Defining the Structure of a JSON Formatted Output Data File | 271 |
| Chapter 7: Setting Up Data Files to Communicate with a Web Service | 272 |
| 7.1 Using an Auxiliary Layout File to Define a Web Service Request Message | 274 |
| 7.2 Using a Reference File or an Initialization File to Define a Request and Receive Data | 275 |
| 7.2.1 Using a Reference File to Define a Request and Receive Data from a SOAP Web Service | 276 |
| 7.2.2 Using a Reference File or an Initialization File to Define a Request and Receive Data from a RESTful Web Service | 279 |
| 7.3 Using a Report File or a Post-Sort Report File to Upload Data | 282 |
| 7.3.1 Using a Report or Post-Sort Report File to Upload Data to a SOAP Web Service | 282 |
| 7.3.2 Using a Report or Post-Sort Report File to Upload Data to a RESTful Web Service | 284 |
| Chapter 8: Setting Up Data Files to Pre-Fill and Mine PDF Forms | 287 |
| 8.1 PDF Form Requirements and Considerations | 288 |
| 8.1.1 Module Requirements | 288 |
| 8.1.2 PDF Form Considerations | 288 |
| 8.2 Pre-Filling Data Fields in PDF Forms | 289 |
| 8.2.1 Specifying the Data to Pre-Fill in a PDF Form | 290 |
| 8.2.2 Placing the Data on a PDF Form in Output | 291 |
| 8.3 Mining Data from PDF Form Fields | 292 |
| 8.3.1 Specifying the Data to Mine from a PDF Form | 293 |
| 8.3.2 Mining Multiple Copies of a PDF Form | 293 |
| Chapter 9: Setting Up Data Files to Mine Data from a Print File | 296 |
| 9.1 Creating a Print File Data File | 297 |
| 9.1.1 Creating a Print File Data File Object | 297 |
| 9.1.2 Defining the Print File Data Format | 297 |
| 9.2 Locating Data in a Print File Data File | 302 |
| 9.2.1 Starting a Page Type, Customer, or Section | 303 |
| 9.2.2 Updating an Existing Page Type | 306 |
| 9.2.3 Setting Up a Spot to Define the Relative Location of Data | 307 |
| 9.3 Mapping a Print File Data Area | 312 |
| 9.3.1 Mapping a Variable to a Data Area | 313 |
| 9.3.2 Specifying How the Engine Uses the Data Area During Processing | 314 |
| 9.3.3 Specifying How the Engine Searches for the Data Area | 316 |

| | |
|---|------------|
| 9.3.4 Specifying the Input Format of Data in a Data Area | 317 |
| Chapter 10: Setting Up Data Files to Extract Data from an ODBC Data Source | 318 |
| 10.1 Supported ODBC Drivers | 319 |
| 10.1.1 Supported Windows ODBC Drivers | 319 |
| 10.1.2 Supported UNIX/Linux ODBC Drivers | 320 |
| 10.1.3 Supported z/OS ODBC Drivers | 321 |
| 10.2 Connecting to an ODBC Data Source | 321 |
| 10.2.1 Creating an ODBC Data Source Data File Object | 322 |
| 10.2.2 Defining the Location of the Data Source For Testing and Production | 322 |
| 10.3 Selecting the Data Source Structure to Map | 324 |
| 10.3.1 Using the Table Query Method to Select Data | 324 |
| 10.3.2 Using the Stored Procedure Method to Select Data | 338 |
| 10.4 Mapping an ODBC Data Source Data File | 342 |
| 10.4.1 Automapping an ODBC Data Source Data File Using Existing Variables .. | 342 |
| 10.4.2 Automapping an ODBC Data Source Data File Using New Variables .. | 343 |
| Chapter 10: Setting up Data Files to Map Data in Communications Designer | 344 |

Chapter 1: Using Data in Exstream Design and Production

To create and produce an application in Exstream Design and Production, you must have data. Data is the key to creating customer output and it is what allows Exstream Design and Production to be dynamic. Without data, the concept of a "customer" wouldn't exist since all of the designs would be the same, with no variation. In order to create an application, you must have data, which is represented by a data file. In order to create a complete application that you can use to produce output, you must include at least one data file in addition to an application object, a document object, and a page object.

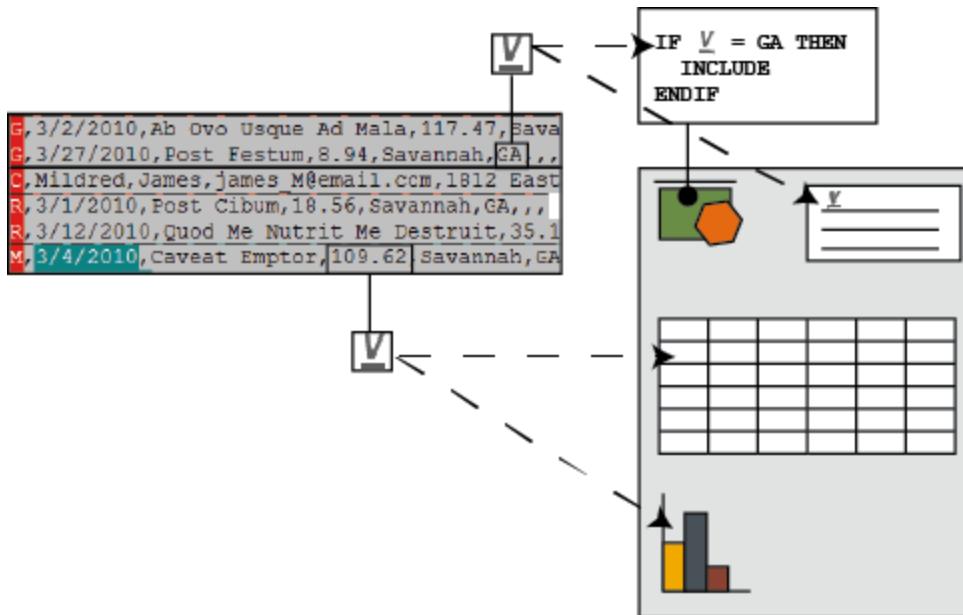
The data you use in Exstream Design and Production can come from multiple locations. For example, you can extract data from a database or you can use data from the design or production system. If you have licensed the appropriate modules, you can also use data directly from a database or from a legacy document. You can store data in any format you want, and if the data is in one of the industry-standard formats supported by Exstream Design and Production, no data consolidation or conversion programs are required. In most cases, you can process the latest available data in your organization before each engine run without preparation.

The data set up process and the application design process go hand-in-hand. Whether the two processes are handled by one person or more than one person, the processes must be coordinated. In order to make sure the correct data is available in the design, as you set up your data, you must consider what data the application requires.

For more information about designing applications, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Data has multiple purposes during the design process. Data is what lets you customize a design for a customer. In fact, you cannot use some design objects, such as charts and some tables, to produce output for a customer without data. Using a variable, you can place customer content from a data file, such as a name, directly in a design. If you use a variable in a rule, you can also use data to select objects based on where a customer lives. During production, the engine takes the design and populates it with the data you have defined. The result is a customized document for each customer.

Data in the application design process



1.1 Data in Exstream Design and Production

To use data in Exstream Design and Production, you must create two types of objects: data files and variables. You use data files to locate and define the customer data used to create an application. Data files also format and define the data written by the engine as it processes an application. Data file objects themselves do not provide data; instead, they define and point to existing data sources created and stored outside of Exstream Design and Production.

Variables hold the place for dynamic content. The content generally comes from three locations: data in a data file, information stored on the design or production system, or information generated during packaging or production. You can use the data from a data file during design to create output customized for each customer. You can use a variable directly on a page to personalize a design using customer data, or in a rule to include or exclude an object from a design or final output. The information stored on the design or production system is useful if you want to include information such as the date in a design or in a report. The information generated during packaging and production lets you create reports about the progress of a production run or send information used for high-volume processing to the production engine.

In order for the engine to use your data to create customized documents, you must complete a process called data mapping. When you map a data file, you associate a variable with a data area. This process creates data maps that the engine uses to find data. The engine then uses the data to populate variables and customize documents. The variable mapped to the data area defines the type of data found in the area. During mapping, you also define how to separate different types of data from each other and the use and format of the data area. For example,

suppose the data in the area is a date. When the engine populates the variable during production, the date in the specified area replaces the variable.

Related information:

- [“Setting Up Data Files” on page 9](#)
- [“Setting Up Variables” on page 59](#)
- [“Data File Mapping” on page 129](#)

1.2 Data-Related Modules

You can expand the data-related capabilities of Exstream Design and Production beyond simple data files and data population in an application. You can add more data formats or additional functionality, such as the ability to leverage legacy files or the ability to send data to or receive data from a DLF file. These features are available if you license one or more of the modules described in the following table.

Data-related modules

| Module | Description |
|------------------|---|
| InteractiveInput | <p>The InteractiveInput module lets you use Live documents (DLF files) as data files to drive data and content extraction, trigger events, or create other documents. It also lets you use DLF files as content to be included in another document and delivered in any format.</p> <p>For more information about using a Live documents as data files, see <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation.</p> |
| ODBC Access | <p>The ODBC Access module lets you use Open Database Connectivity (ODBC) to directly access data in a data source. It lets you access and map variables to the contents of ODBC relational databases without building custom programs to extract the data or mapping variables to flat files.</p> <p>For more information about using an ODBC data source, see “Setting Up Data Files to Extract Data from an ODBC Data Source” on page 318.</p> |
| PDF Form Miner | <p>The PDF Form Miner module lets you integrate PDF XFA forms into the Exstream Design and Production application creation process. It lets you extract data contained in a PDF XFA form and use it as input to drive the personalization and production of an application in Exstream Design and Production.</p> <p>Note: The PDF Form Miner module is not supported on z/OS.</p> <p>For more information about using a PDF form, see “Setting Up Data Files to Pre-Fill and Mine PDF Forms” on page 287.</p> |

Data-related modules, continued

| Module | Description |
|------------------------|---|
| PDF Form Pre-Fill | <p>The PDF Form Pre-fill module lets you integrate Exstream Design and Production into the PDF XFA form creation process. It lets you customize PDF XFA forms by automatically populating some or all of the fields in a pre-existing form with customer data stored in an external source.</p> <p>Note: The PDF Form Pre-Fill module is not supported on z/OS.</p> <p>For more information about using a PDF form, see “Setting Up Data Files to Pre-Fill and Mine PDF Forms” on page 287.</p> |
| Print Miner | <p>The Print Miner module lets you leverage the content in archived or legacy print files without making changes to your legacy documents or archival systems. The Print Miner module lets you mine the data from the archived documents just as if they were flat data files.</p> <p>For more information about using a print file, see “Setting Up Data Files to Mine Data from a Print File” on page 296.</p> |
| Web Services Interface | <p>The Web Services Interface module lets you create output using data delivered by a web service during production and to store requested data from a web service, without additional web service setup on the target production platform. If you are working in a Service-Oriented Architecture (SOA) based organization, it also lets you leverage your existing application infrastructure.</p> <p>For more information about using a web services file, see “Setting Up Data Files to Communicate with a Web Service” on page 272.</p> |
| XML/JSON (Data) Output | <p>The XML/JSON (Data) Output module lets you create XML or JSON formatted report files that record information about both one-step and two-step production runs.</p> <p>For more information, see “Setting Up XML Formatted Data Files” on page 178 and “Setting Up JSON Formatted Data Files” on page 253.</p> |
| XML/JSON Input | <p>The XML/JSON Input module lets you use XML or JSON formatted data files to drive the creation of applications. The data files can be used for both one-step and two-step processing.</p> <p>For more information, see “Setting Up XML Formatted Data Files” on page 178 and “Setting Up JSON Formatted Data Files” on page 253.</p> |

1.3 Section Data in Exstream Design and Production

The layout of data contributes to decisions such as how to create and map the data file, how to calculate the value of a variable, and how to time the inclusion of design objects during production. One kind of data layout is called section data. Section data is a data layout that uses groups of records related to one particular type of information, such as all of the debits or all of the deposits for an account.

Section data example

| | |
|---------|---|
| Section | G, 3/2/2010, Ab Ovo Usque Ad Mala, 117.47, Savannah, GA,, |
| | G, 3/27/2010, Post Festum, 8.94, Savannah, GA,,, |
| | C, Mildred, James, james.M@email.com, 1812 East |
| Section | R, 3/1/2010, Post Cibum, 18.56, Savannah, GA,,, |
| | R, 3/12/2010, Quod Me Nutrit Me Destruct, 35.1 |
| | M, 3/4/2010, Caveat Emptor, 109.62, Savannah, GA |

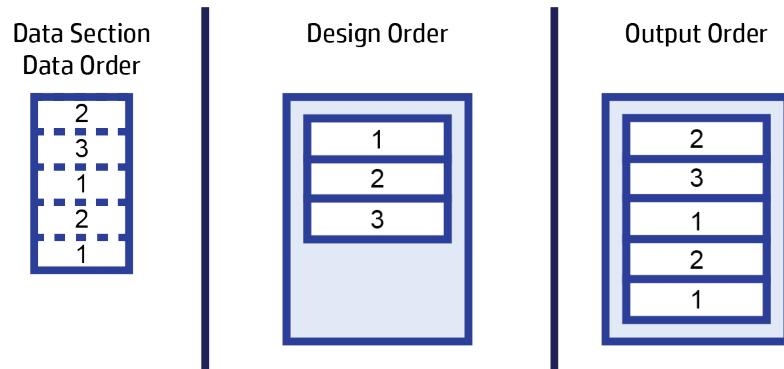
Since it helps identify groups of similar data, section data is especially useful when the order of the data is different for each customer, or when the number of data types varies for each customer. Section data is also useful in documents that are similar in design but that contain different types of data, and for transactional documents (such as account statements). Section data lets you design a document, table, or section object that can be used with multiple sets of data in a customer, particularly in transactional documents or in householded mailings.

For more information about using section data to create a document, a table, or a section object, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Before you create variables and data files, and before you create section-driven objects in your design, consider how the engine processes section data. When the engine processes section data, it reads a section at a time and only the data within that section is available, unless data is stored in temporary variables. Usually, the design order of an object determines the order in which it is placed in the output. However, with section data, because the variables reset each time the engine encounters a new data section, the data order determines the order in which the object is placed in the output. For example, if you design a table with multiple sections, the data determines the order in which the table sections are placed on the page, even if the table sections are in a different order in the design.

For more information about engine processing, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

Section data processing order



You can use section data in combination with rules to further customize the object order. Keep in mind, though, that the engine includes objects with a rule when it composes the document,

which occurs before section processing. Because these inclusion methods have different timing, rule-based objects appear in the output before section data-based objects.

For more information about rules, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Section data can introduce additional complexity to a document, which can increase processing time. Do not use section data simply because the data is repetitive. If all of the records in a customer act as a single group, adding a section for each record reduces performance and data availability. Section data is also unnecessary if you filter the data in the design. For example, suppose that you map an 'AccountType' variable to a data area that includes both checking and savings data. If you place a rule on a table row so that only checking data is included, section data is not required.

Keep in mind that you cannot use section data with an ODBC data source data file.

1.4 XML Nodes in Exstream Design and Production

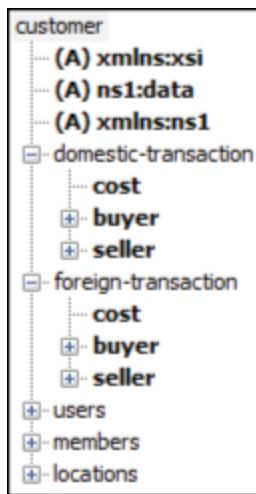
Note: XML nodes are accessible only in schema model data files.

For more information about schema model data files, see "[Setting Up XML Formatted Data Files](#)" on page 178.

In XML schema-based files, such as the schema model data file, the data layout is based on the original XML hierarchy and on the built-in structure of the XML file. In this layout, each element within the data is considered to be an XML node, and through the hierarchy, the nodes form the structure of the data.

Because all of the content within a schema-based file is naturally represented as XML nodes in the original XML hierarchy, you can connect application objects with XML nodes in the structure and drive the inclusion of the objects in the application directly from the XML structure, rather than rely on data sections.

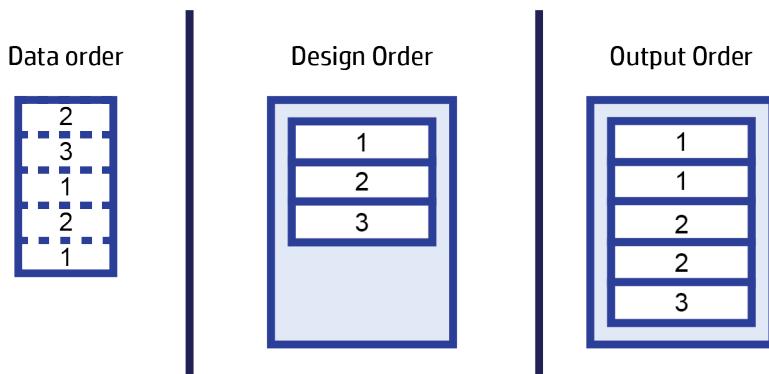
Example of a schema model data file with XML nodes (XML Schema Browser view)



Before you create variables and data files, and before you design an application that is driven with XML nodes, consider how the engine processes XML nodes. Schema model data files are processed by the engine during the data aggregation engine timing, in order to gain broader access to customer data. During the data aggregation engine timing, the engine has access to all of the customer data at one time. This process allows the engine to read all of the customer data, including any XML nodes, and to hold the data in memory until it reaches the next customer.

Each time the engine encounters XML node-driven content in the application, the engine queries the customer data for any additional matching XML nodes. For example, if you have a document that is driven by a SAVINGS node in the schema model data file, and there are multiple occurrences of the SAVINGS node in the data, the engine will also compose all other instances of the document that are driven based on the SAVINGS node. Because of this processing order, objects that are driven by the same XML nodes are rendered successively by the engine, even if the node order does not appear successively in the data source. Keep in mind that this output order can be impacted by the hierarchy of your data source and whether you are using multiple XML node-driven objects in the same application.

Section data processing order



For more information about engine processing, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

You can use XML nodes in combination with rules to further customize the object order. Keep in mind, though, that the engine includes objects with a rule when it composes the document, which occurs before XML nodes-driven objects are processed by the engine. Because these inclusion methods have different timing, rule-based objects appear in the output before XML node-based objects.

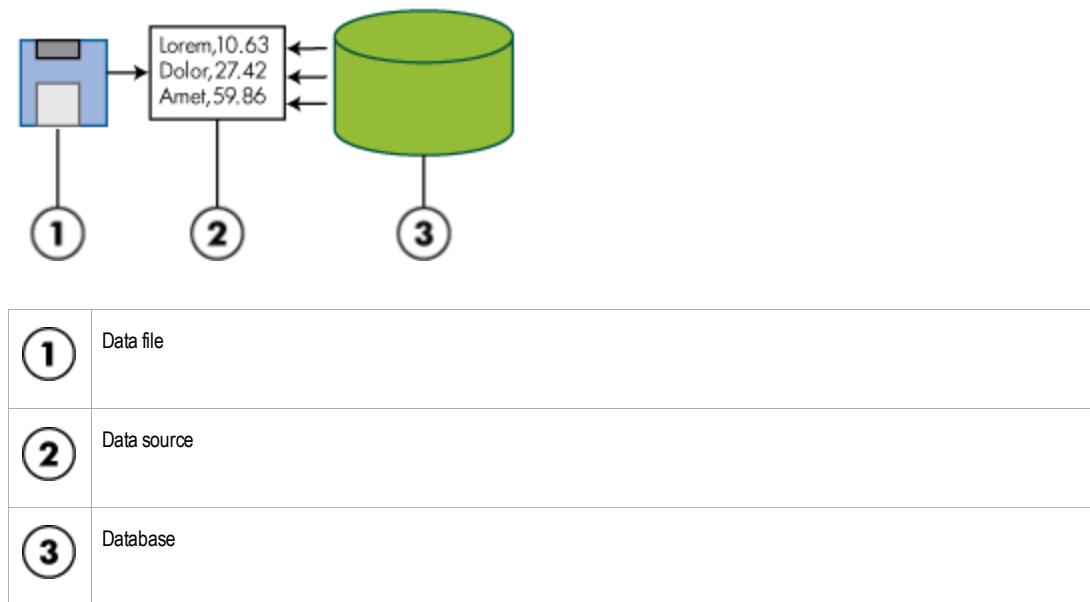
For more information about rules, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Chapter 2: Setting Up Data Files

Data files are the key to creating and customizing output. In Exstream Design and Production, you use data files to locate and define the customer data used to create an application. Data files also format and define the data written by the engine as it processes an application.

Data file objects themselves do not provide data; instead, they define and point to existing data sources created and stored outside of Exstream Design and Production. The data sources you use in Exstream Design and Production can come from multiple locations. In most cases, a data source is a file that you have extracted from a database. This process helps you ensure that no accidental changes are made to the data before or during processing. As shown in the following graphic, the data source is extracted from the database and the data file points to data source, not the database.

Data file pointing to a data source



After you have created data files, you can associate variables with a portion of the data file through a process called mapping. When you map a data file, you associate a variable with a data area in the data file. This process creates data maps that the engine uses to find data. The engine then uses the data to populate variables and customize documents. For example, after you map a variable to a data file, you can use the customer information represented by the variable to place content directly on a page, in a rule to select objects, or to order documents during production.

For more information about variables, see [“Setting Up Variables” on page 59](#).

For more information about mapping a data file, see [“Data File Mapping” on page 129](#).

Depending on how your organization uses and creates data files, you can create a single data file that you can use in multiple applications, or you can use the same data sources in two

different data files. For example, suppose that you have a correspondence application and a statement application. If the two applications require similar data, you can create a single data file that you include in both applications. If the correspondence application requires only a portion of the data required for the statement application, you can use the same data source, but map only the data used by the correspondence application.

This chapter discusses the following topics:

- “[Data File Formats and Types](#)” below
- “[Parts of a Data File](#)” on page 16
- “[Creating an Auxiliary Layout Data File](#)” on page 18
- “[Creating a Customer Driver Data File](#)” on page 21
- “[Creating an Initialization or a Post-sort Initialization Data File](#)” on page 29
- “[Creating a Reference Data File](#)” on page 37
- “[Creating a Report or a Post-sort Report File](#)” on page 50

2.1 Data File Formats and Types

You can categorize data files based the layout of the data to which they point or based on the kind of information they provide to the engine. The data file format defines the layout of the data. For example, the data file can point to a data source that organizes data using columns or to a data source that uses XML tags to surround data. The data file type defines the kind of information the data file provides the engine. For example, the data file contains customer information, or it provides the layout for a report that the engine will generate when it runs.

This section discusses the following topics:

- “[Data File Formats](#)” below
- “[Data File Types](#)” on page 12
- “[Data File Type and Format Availability](#)” on page 14

2.1.1 Data File Formats

Data file formats define the layout of data in a data file. You view this layout when you open a data file in the Edit Panel. Exstream Design and Production supports multiple data file formats so that the engine can read and write customer data in the format best suited for your data source, or other considerations, such as your archival system. You can mix multiple formats in a single application. The following table lists the available formats.

Available data file formats

| Format | Description |
|------------------|---|
| Columnar | <p>The data file consists of data organized into columns.</p> <p>This data file format is available by default. You are not required to license an additional module.</p> |
| Delimited | <p>The data file consists of data separated by a delimiter character, such as a comma. The delimiters define where one data element begins and another one ends.</p> <p>This data file format is available by default. You are not required to license an additional module.</p> |
| JSON | <p>The data file consists of data that uses the JavaScript Object Notation (JSON) text format to define data elements.</p> <p>To use this data file format:</p> <ul style="list-style-type: none">• To create JSON formatted input data files, you must have licensed the JSON Input module.• To create JSON formatted report files, you must have licensed the JSON (Data) Output module.• To create JSON formatted data files that can be used to communicate with RESTful web services, you must have licensed the Web Services Interface module, the Empower Output module, or the DLF module. <p>For more information about using a JSON file, see "Setting Up JSON Formatted Data Files" on page 253.</p> |
| Live | <p>The data file consists of data located in a Live Document (DLF) file. You use XML to map the data you want to use.</p> <p>This data file format is available only if you have licensed the InteractiveInput module.</p> <p>For more information about using a Live file, see <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation.</p> |
| ODBC data source | <p>The data file consists of data located directly in a database. The data has not been extracted from an external data source; instead, the data is accessed directly from the database using SQL queries.</p> <p>This data file format is available only if you have licensed the ODBC Access module.</p> <p>For more information about using an ODBC data source, see "Setting Up Data Files to Extract Data from an ODBC Data Source" on page 318.</p> |
| PDF form | <p>The data file consists of data located on the XML layer of a PDF XFA form.</p> <p>To use this data file format:</p> <ul style="list-style-type: none">• To add data to a PDF form, you must have licensed the PDF Form Pre-fill module.• To mine data from a PDF form, you must have licensed the PDF Form Miner module <p>For more information about using a PDF form, see "Setting Up Data Files to Pre-Fill and Mine PDF Forms" on page 287.</p> |
| Printfile | <p>The data file consists of data located in a print file. You mine the print file in order to leverage data without making changes to your legacy documents or archival systems.</p> <p>This data file format is available only if you have licensed the Print Miner module.</p> <p>For more information about using a print file, see "Setting Up Data Files to Mine Data from a Print File" on page 296.</p> |

Available data file formats, continued

| Format | Description |
|-------------------------|--|
| XML | <p>The data file consists of data that uses tags to define the beginning and end of a data element.</p> <p>To use this data file format:</p> <ul style="list-style-type: none">• To create XML formatted input data files, you must have licensed the JSON Input module.• To create XML formatted report files, you must have licensed the JSON (Data) Output module.• To create XML formatted data files that can be used to communicate with RESTful or SOAP web services, you must have licensed the Web Services Interface, Empower, or DLF modules. <p>For more information about using an XML file, see "Setting Up XML Formatted Data Files" on page 178.</p> |
| Communications Designer | <p>The data file is used in the package file as a placeholder for the Communications Designer data source.</p> <p>This data file format is available only if you have licensed the Communications Designer module.</p> <p>For more information about using a Communications Designer data source, see "Setting up Data Files to Map Data in Communications Designer" on page 344.</p> |

2.1.2 Data File Types

Data file types define the kind of information the data file provides the engine. It also defines when the information is available to the engine. For example, some data file types set variable values before the engine reads customer data and other data file types write data as the engine processes an application. Exstream Design and Production supports multiple data file types in a single application. This ability helps ensure that the engine can access all of the data the application needs at the correct time. The following table lists the available data file types.

Data file types

| Data file type | Icon | Description |
|-----------------------|------|---|
| Auxiliary layout file | | Auxiliary layout data files let you create data mappings that can subsequently be associated with another data file. For example, you can use an auxiliary layout file in order to use more than one variable in a customer driver file as a key to access reference files. |
| Customer driver file | | Customer driver files contain at least one set of records for each customer. They drive the main processing of the engine and are usually the primary source of data. |
| File viewer | | Unlike other data file types, file viewer data files do not let you associate variables with data. However, like other data file types, they let you view content in the Edit Panel. File viewer data file types support Metacode and AFP output. For more information about sort index files, see <i>Creating Output</i> in the Exstream Design and Production documentation. |
| Initialization file | | Initialization files set initial values for specified variables at the beginning of the production run. These variables are not customer dependent. |

Data file types, continued

| Data file type | Icon | Description |
|-------------------------------|---|--|
| Post-sort initialization file |  | <p>Post-sort initialization files serve the same function as an initialization file. However, the engine reads them only during the second pass of two-step processing.</p> <p>For more information about one-step and two-step processing, see <i>Preparing Applications for Production</i> in the Exstream Design and Production documentation.</p> |
| Post-sort report file |  | <p>Post-sort report files serve the same function as a report file. However, the engine reads them only during the second pass of two-step processing.</p> <p>For more information about one-step and two-step processing, see <i>Preparing Applications for Production</i> in the Exstream Design and Production documentation.</p> |
| Reference file |  | <p>Reference files let the engine access customer information that is not included in a customer driver file.</p> |
| Report file |  | <p>Report files let the engine collect data in order to create a report that includes any information you select about the production run.</p> |
| Signature report file |  | <p>Signature report files are used to support the use of electronic signatures in PDF output. This type of file is available when you have licensed the Electronic Signature Integration module.</p> <p>For information about using electronic signatures in an application, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> |
| Sort index file |  | <p>Sort index files let you drive the order of processing during the second pass of two-step processing. On the first pass, the engine creates the sort index, and then in the second pass, the engine uses the sort index file to order the processing.</p> <p>For more information about one-step and two-step processing, see <i>Preparing Applications for Production</i> in the Exstream Design and Production documentation.</p> <p>For more information about sort index files, see <i>Creating Output</i> in the Exstream Design and Production documentation.</p> |

When you are [setting up a data file to communicate with a web service](#), you can also select the file source type to define the type of web service you want to use to deliver data during production.

Keep in mind that the **File source type** list is displayed only when you select a data file type and file format combination that can be used to set up a data file to communicate with a web service.

The following table lists the available data file source types.

Available data file source types

| Data file source type | Description |
|-----------------------|---|
| File | The data file will not be used with a web service during production. This option is selected by default when the File source type list is displayed. |

Available data file source types, continued

| Data file source type | Description |
|-----------------------|---|
| Web Service (RESTful) | <p>The data file consists of data delivered by a RESTful web service during production. The data can also be sent to a RESTful web service for archiving, notification purposes, or subsequent processing. You use JSON or XML to map the data you want to use. This data file format is available only if you have licensed the Web Services Interface module, the Empower Output module, or the DLF module.</p> <p>For more information about using a web services file, see “Setting Up Data Files to Communicate with a Web Service” on page 272.</p> |
| Web Service (SOAP) | <p>The data file consists of data delivered by a SOAP web service during production. The data can also be sent to a SOAP web service for archiving, notification purposes, or subsequent processing. You use XML to map the data you want to use. This data file format is available only if you have licensed the Web Services Interface module, the Empower Output module, or the DLF module.</p> <p>For more information about using a web services file, see “Setting Up Data Files to Communicate with a Web Service” on page 272.</p> |

During a production run, the engine processes the data files in the order you place them in the application. To make sure that data files are accessible to the engine at the correct time, you must place them in an application in the following order:

1. Initialization file
2. Customer driver files (in the order the engine accesses them)
3. Reference files (in the order the engine accesses them)
4. Auxiliary layout files
5. Report files (must be last)

2.1.3 Data File Type and Format Availability

The data file type, file format, and file source type work together to let you gather and use data in a manner that best fits your processes and tools. Not all file formats and file source types are available for all data file types, though.

The following table lists the data file types, file formats, and file source types that are available for each data file type.

Data file formats per data type

| File type | Supported file formats | Supported file source types |
|-------------------------------|---|---|
| Auxiliary layout file | <ul style="list-style-type: none">• Columnar• Delimited• JSON• ODBC data source• PDF form• XML | <ul style="list-style-type: none">• File• Web Service (RESTful) for JSON and XML file formats• Web Service (SOAP) for XML file format |
| Customer driver file | <ul style="list-style-type: none">• Columnar• Delimited• Live• JSON• ODBC data source• PDF form• Print file• XML• Communications Designer | File |
| Initialization file | <ul style="list-style-type: none">• Columnar• Delimited• JSON• ODBC data source• PDF form• XML | <ul style="list-style-type: none">• File• Web Service (RESTful) for JSON and XML file formats |
| Post-sort initialization file | <ul style="list-style-type: none">• Columnar• Delimited• ODBC data source• PDF form• XML | File |

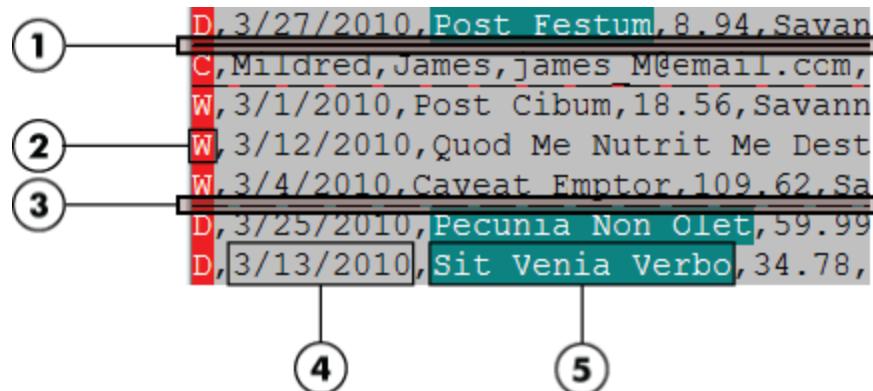
Data file formats per data type, continued

| File type | Supported file formats | Supported file source types |
|-----------------------|--|---|
| Post-sort report file | <ul style="list-style-type: none">• Columnar• Delimited• ODBC data source• PDF form• XML | <ul style="list-style-type: none">• File• Web Service (RESTful) for XML file formats• Web Service (SOAP) for XML file format |
| Reference file | <ul style="list-style-type: none">• Columnar• Delimited• JSON• Live• ODBC data source• PDF form• Printfile• XML | <ul style="list-style-type: none">• File• Web Service (RESTful) for JSON and XML file formats• Web Service (SOAP) for XML file format |
| Report file | <ul style="list-style-type: none">• Columnar• Delimited• JSON• ODBC data source• PDF form• XML | <ul style="list-style-type: none">• File• Web Service (RESTful) for JSON and XML file formats• Web Service (SOAP) for XML file format |
| Signature report file | <p>XML</p> <p>For information about using signature report files, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> | File |

2.2 Parts of a Data File

Data files can be broken up into customers, sections, records, and data areas. A data area is a single piece of data within the data file. A record is a series of data areas, typically organized in a row, that are similar in nature. Sections are groups of records related to one particular type of information, such as all of the debits or all of the deposits for an account. Customers are groups of records and sections related to a single customer. You can identify customers, sections, and records using record type indicators. As shown in the following graphic, these various parts work together to define the layout of data in a data file.

Parts of a data file in the Edit Panel



| | |
|----------|-----------------------|
| 1 | Customer start |
| 2 | Record type indicator |
| 3 | Section start |
| 4 | Unmapped data area |
| 5 | Mapped data area |

2.2.1 Record Type Indicators

Record type indicators are special parts of a data file that you can use to identify the beginning of customers, sections, and records. Typically, you use numbers as record type indicators, but you can also use letters, spaces, nulls, and most non-printing characters. Records with the same record type indicator have the same type of data. When the record type indicator changes, the type of data changes as well.

You can also use sub-indicators to accommodate conditional sections. For example, if the data file contains transactional data for different types of accounts, each customer's data might be in the 80 record. Within the 80 record, savings account data is in the S record, checking account data is in the C record, and ATM data is in the A record.

Record indicators have the following specifications:

- The maximum length of a record indicator is 50 characters.
- Records can have multiple indicator locations.

- Each section can contain any number of records and any number of sub-sections with their own record indicators.

2.3 Creating an Auxiliary Layout Data File

Auxiliary layout data files are optional data files that let you create data mappings that can subsequently be associated with another data file. For example, you can use an auxiliary layout file in order to meet goals such as the following:

- Use more than one variable in a customer driver file as a key to access reference files.
- Format data being sent to an external location, such as a web service.
- Provide transaction information for use with the MapLayout function.

If you use an auxiliary layout file, you must order it after the reference files in the application. You cannot add rules to auxiliary layout files. Unlike other data file types, which are used for input or output only, auxiliary layout files can be used as both input and output files.

To create an auxiliary layout file, you must complete the following tasks:

1. [“Creating an Auxiliary Layout File Object” on the next page](#)
2. [“Defining the Auxiliary Layout Format” on page 20](#)

2.3.1 Creating an Auxiliary Layout File Object

1. To select an application mode, complete one of the following tasks:

| To | Do this |
|--|---|
| Create an auxiliary layout data file in SBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create an auxiliary layout data file in DBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create an auxiliary layout data file in SBCS/DBCS mode | a. In Design Manager, in the Library, expand the Data Files heading. b. In the Library, right-click the type of data file you wish to create. For example, select SBCS data file for a new SBCS data file. The New Data File dialog box opens. |

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Auxiliary layout**.
4. To specify the format of the data source, select one of the following options from the **File format** list:
 - **Columnar**
 - **Delimited**
 - **JSON**
 - **ODBC**
 - **PDF Form**
 - **XML**
5. To specify the source of the data, select one of the following options from the **File source type** list:
 - **File**
 - **Web Service (RESTful)**
 - **Web Service (SOAP)**

Note: The Web Service (SOAP) source is not available for the JSON file format.

6. Click **Finish**.

The **New Data File** dialog box closes and the auxiliary layout file opens in the Property Panel for you to define.

For more information about the available data file formats, see “[Data File Formats and Types](#)” on [page 10](#).

Note: If you choose to create a new auxiliary data file by selecting **Insert > Data File**, you must specify the application mode from the **Application mode** list in the **New Data File** dialog box.

2.3.2 Defining the Auxiliary Layout Format

When you define the format of an auxiliary layout file, you define the format of the data source and the layout of the data file.

To define the auxiliary layout file format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. For all formats, other than ODBC, in the **Sample layout** box, enter the external layout file that defines the layout of the data.
4. To set up format-specific formatting, do one of the following, depending on the format you selected:

| If you selected this format | Do this |
|-----------------------------|---|
| Columnar | Continue to step 5. |
| Delimited | <ol style="list-style-type: none">a. In the Delimiter box, enter the character used to separate data.<p>Note: For best results, OpenText recommends that you use delimiters from the standard ASCII printable range (32–126) for SBCS data files.</p>b. If you want the delimiter to appear in hexadecimal code, select the Show hex check box.c. If the delimited data contains quoted fields, you must do the following:<ol style="list-style-type: none">i. Select the Fields may be quoted with check box.ii. In the adjacent box, enter the character used to surround data information. The engine ignores this character when it processes the data. |

| If you selected this format | Do this |
|-----------------------------|--|
| JSON | Complete the steps described in “Setting Up JSON Formatted Data Files” on page 253. If you are using the auxiliary layout to communicate with a web service, see “Setting Up Data Files to Communicate with a Web Service” on page 272. |
| ODBC | Complete the steps described in “Setting Up Data Files to Extract Data from an ODBC Data Source” on page 318. |
| PDF Form | Complete the steps described in “Setting Up Data Files to Pre-Fill and Mine PDF Forms” on page 287. |
| XML | Complete the steps described in “Setting Up XML Formatted Data Files” on page 178. If you are using the auxiliary layout to communicate with a web service, see “Setting Up Data Files to Communicate with a Web Service” on page 272. |

5. From the **Edit** menu, select **Save**.

2.4 Creating a Customer Driver Data File

Customer driver files are the only type of data files required to package an application. Customer driver files contain at least one set of records for each customer. They drive the main processing of the engine and are usually the primary source of data.

Unless you include an initialization file in an application, the customer driver file must be the first data file in the application. If an application has multiple customer driver files, the engine processes them in the order they appear under the application in the Library.

To create a customer driver file, you must complete the following tasks:

1. [“Creating a Customer Driver File Object” on the next page](#)
2. [“Defining the Customer Driver File Format” on page 23](#)
3. [“Defining the Records in a Customer Driver File” on page 25](#)
4. [“Selecting the Customer Driver File Data Source” on page 27](#)

2.4.1 Creating a Customer Driver File Object

1. To select an application mode, complete one of the following tasks:

| To | Do this |
|--|---|
| Create a customer driver data file in SBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a customer driver data file in DBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a customer driver data file in SBCS/DBCS mode | a. In Design Manager, in the Library, expand the Data Files heading. b. In the Library, right-click the type of data file you wish to create. For example, select SBCS data file for a new SBCS data file. The New Data File dialog box opens. |

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Customer driver file**.
4. To specify the format of the data source, select one of the following options from the **File format** list:
 - **Columnar**
 - **Delimited**
 - **Interactive**
 - **JSON**
 - **ODBC**
 - **PDF Form**
 - **Print**
 - **XML**
 - **Schema Model**
 - **Communications Designer**
5. Click **Finish**.

The **New Data File** dialog box closes and the customer driver file opens in the Property Panel for you to define.

For more information about the available data file formats, see “[Data File Formats and Types](#)” on [page 10](#).

Note: If you choose to create a new customer driver data file by selecting **Insert > Data File**, you must specify the application mode from the **Application mode** list in the **New Data File** dialog box.

2.4.2 Defining the Customer Driver File Format

When you define the format of a customer driver file, you define the layout of the data and the format of the characters in the data source. You also define the data so that you can correctly map the data file in the Edit Panel.

To define the customer driver file format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To set up specific formatting, do one of the following, depending on the file format you selected:

| If you selected this format | Do this |
|-----------------------------|--|
| Columnar | Continue to step 4. |
| Delimited | <ol style="list-style-type: none">a. In the Delimiter box, enter the character used to separate data.<p>Note: For best results, OpenText recommends that you use delimiters from the standard ASCII printable range (32–126) for SBCS data files.</p>b. If you want the delimiter to appear in hexadecimal code, select the Show hex check box.c. If the delimited data contains quoted fields, you must do the following:<ol style="list-style-type: none">i. Select the Fields may be quoted with check box.ii. In the adjacent box, enter the character used to surround data information. The engine ignores this character when it processes the data. |
| JSON | Complete the steps described in “ Setting Up JSON Formatted Data Files ” on page 253 . |
| Live | Complete the steps described in <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation. |
| ODBC | Complete the steps described in “ Setting Up Data Files to Extract Data from an ODBC Data Source ” on page 318 . |

| If you selected this format | Do this |
|-----------------------------|--|
| PDF Form | Complete the steps described in "Setting Up Data Files to Pre-Fill and Mine PDF Forms" on page 287 . |
| Print | Complete the steps described in "Setting Up Data Files to Mine Data from a Print File" on page 296 . |
| XML | Complete the steps described in "Setting Up XML Formatted Data Files" on page 178 . |

4. Go to the **Advanced** tab.
5. To define the character format of the data source, do one of the following, depending on the type of application in which you are working:

| For this type of application | Do this |
|------------------------------|--|
| SBCS | From the Character set list, select one of the following: <ul style="list-style-type: none">• Native—The engine determines the format at the data source automatically.• ASCII—The data source contains data in ASCII.• EBCDIC—The data source contains data in EBCDIC or it contains z/OS-specific formats such as packed decimal or zoned. If you are using EBCDIC files in an ASCII environment, the files must have a CR/LF (x0d0a) at the end of each record.• UTF-8—The data source is an XML or JSON file that uses UTF-8 encoding. This option is available only if you have licensed the XML/JSON Input module. |
| DBCS | From the Encoding box, select the encoding of the characters in the data source. The options vary based on the encodings enabled on your system. |

6. If the data source originates from a platform different from the one being used to map the data file, select one of the following options from the **Binary integer byte order** list in order to define the order of the bytes in the data source:
 - **Native**—Use the original byte order.
 - **Big-endian**—Convert the file to big-endian order. The first time you run Design Manager, it automatically converts existing EBCDIC data files to big-endian order.
 - **Little-endian**—Convert the file to little-endian order.

Note: Exstream Design and Production does not support data files with packed binary data for DBCS applications.

7. From the **Edit** menu, select **Save**.

2.4.3 Defining the Records in a Customer Driver File

When you define the records in a customer driver file, you define how the data file separates the records of one customer from another customer and how the data file separates one individual record from another record. You also define which records you want to view and which ones are ignored when producing output.

To define records in a customer driver file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the method in which customer records are separated at the data source, do one of the following:

| To | Do this |
|---|---|
| Specify that all customers have the same number of records (Columnar and Delimited) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select Each customer has the same number of records.b. In the Number of records per customer box, enter the number of records per customer. |
| Specify that the file uses a single record type to indicate new customers (Columnar) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A new customer occurs on specified record type(s).b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that a new customer starts each time the record type indicator changes (Columnar) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A customer ID is on every record.b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that the file uses a single record type to indicate new customers (Delimited) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A new customer occurs on specified record type(s).b. In the Delimited field index of the first record-type indicator (1-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |

| To | Do this |
|--|--|
| Specify that a new customer starts each time the record type indicator changes (Delimited) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A customer ID is on every record.b. In the Delimited field index of the first record-type indicator (1-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |

4. If you are using indicators, and your data file has sub-indicators, you must do the following:
 - a. Select the **Multiple indicators** check box.
The **Edit Record Indicators** dialog box opens.
 - b. In the **Start** box, enter the start location.
 - c. In the **Length** box, enter the length of the sub-indicator.
 - d. Click **+**.
 - e. Click **OK**.
5. Go to the **Advanced** tab.
6. To specify how the data file separates one record from the next record, select one of the following options from the **Record type** list:
 - **CR/LF or LF**—Records end with a carriage return (CR) and line feed (LF), or an LF only. If you select this option and have licensed the On Demand Delivery module, the engine skips bad data records based on CR/LFs or LFs. If the bad data does not contain a CR/LF or LF, the engine combines it with records from the next data file until it encounters a CR/LF or LF in the read buffer.
 - **Fixed Length**—Records are the same length. In the **Length** box, enter the length of the records.
 - **Blocked (prefixed w/ 2 byte big-endian, exclusive)**—Records are preceded by a two-byte big-endian exclusive record length, which indicates the length of the record. This option is applicable only in the z/OS environment.
 - **CR/LF only**—Records end with a CR and an LF.
 - **MVS FTP block mode (prefixed w/ x80 + 2 bytes)**—Records are automatically reblocked for FTP. This option is applicable only in the z/OS environment.
 - **Blocked (prefixed w/ 4 byte big-endian, inclusive)**—Records are preceded by a four-byte big-endian inclusive record (two bytes of length followed by two null bytes). This option is applicable only in the z/OS environment.
7. To limit the number of records visible in the Edit Panel (for memory constraints or visual preference, for example) enter the number of records in the **Limit datamapping records**

box. The default value is 1000. Enter 0 to specify that there are no constraints.

8. To set initial data without using an initialization file, enter the number of records the headers occupy in the file in the **Header** box. When the engine runs, information in the header records (including the validation record) is not produced as output. During mapping, make sure the variables mapped to the header records have a setting of **Never reset** in the **Reset time** list.

For more information about variable properties, see “[Setting Up Variables](#)” on page 59.

Tip: If you are unsure of the exact number of header records in the data file, you can set the number higher than the actual number of header records. When the engine encounters a record specifying a customer, it stops reading header records.

9. From the **Edit** menu, select **Save**.

2.4.4 Selecting the Customer Driver File Data Source

When you select the data source for a customer driver file, you define one data source for mapping and testing and another data source for production. You map the test data source. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. The test data source is not the data source the engine uses to produce applications, however. During production, the engine uses the data maps, along with the production data source, to read and write customer data. The test data source and the production data source share many of the same properties to make sure your tests are as accurate as possible.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

To select the customer driver file data source:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Advanced** tab.
3. To assign a connector that points to a routine that transforms incoming data before it reaches the engine, do one or both of the following steps based on when you want to use the connector.

| To | Do this |
|-------------------------------------|--|
| Use the connector during testing | From the Test record transform box, select an existing connector object from the Library. |
| Use the connector during production | From the Production record transform box, select an existing connector object from the Library. |

4. If you want the engine to verify that it is reading the correct source data, you must do the following:
 - a. Make sure the source data has a character string at the beginning that the engine can compare with the string you enter in the **Validation Record** box.
 - b. In the **Validation Record** box, enter the text that must match the string in the data source. The engine reads only the first 16 characters of the string.
5. If data for a customer can change during processing (you are using a connector to connect to the data source, for example) and you want to update the information in the data source, you must do the following:
 - a. Select the **File can be updated** check box.
 - b. In the **Update output file name when running in test mode** box, enter the path to the output file.
 - c. In the **Update output file name when running in production mode** box, enter the path to the output file.

Note: Use the Trigger function to control the timing of the update operation.

6. In the **Rule** box, enter a rule to include or exclude the file during an engine run.
7. On the **Test Data Source** tab, you must do one of the following to specify the location of the data source to use for testing and mapping:

| To | Do this |
|--|--|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use for data mapping box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use for data mapping box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

8. On the **Production Data Source** tab, you must do one of the following to specify the location of the data source to use during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use for data mapping box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p></div> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the Connector box, select an existing connector object from the Library. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p></div> |

9. From the **Edit** menu, select **Save**.

2.5 Creating an Initialization or a Post-sort Initialization Data File

Initialization files are optional files that the engine reads before any other customer data. Initialization files set initial values for specified variables at the beginning of the production run, instead of each time the variables occur. These variables are not customer dependent. Post-sort initialization files serve the same function as an initialization file. However, the engine reads them only during the second pass of two-step processing. If you include an initialization file in an application, it must be the first data file in the application.

The engine holds all the data in an initialization file in memory until the engine completes processing. If you load a large amount of data into memory, you decrease processing efficiency. In cases of large initialization files, try using a reference file.

For more information about setting up a reference file, see “[Creating a Reference Data File](#)” on page 37.

To create an initialization or a post-sort initialization file, you must complete the following tasks:

1. [“Creating an Initialization or a Post-sort Initialization File Object” below](#)
2. [“Defining the Initialization or Post-sort Initialization File Format” on the next page](#)
3. [“Defining the Records in an Initialization or a Post-sort Initialization File” on page 33](#)
4. [“Selecting the Initialization or Post-sort Initialization File Data Source” on page 35](#)

2.5.1 Creating an Initialization or a Post-sort Initialization File Object

1. To select an application mode, complete one of the following tasks:

| To | Do this |
|--|--|
| Create an initialization or post-sort initialization data file in SBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create an initialization or post-sort initialization data file in DBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create an initialization or post-sort initialization data file in SBCS/DBCS mode | <ol style="list-style-type: none">a. In Design Manager, in the Library, expand the Data Files heading.b. In the Library, right-click the type of data file you wish to create. For example, select SBCS data file for a new SBCS data file. The New Data File dialog box opens. |

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select one of the following options:
 - **Initialization file**
 - **Post-sort initialization file**.
4. To specify the format of the data source, select one of the following options from the **File format** list:
 - **Columnar**
 - **Delimited**
 - **JSON**
 - **ODBC**

- **PDF Form**
- **XML**

Note: The JSON format is not available for post-sort initialization files.

5. To specify the source of the data, select one of the following options from the **File source type** list:
 - **File**
 - **Web Service (RESTful)**

Note: The Web Service (RESTful) source is not available for post-sort initialization files.

6. Click **Finish**.

The **New Data File** dialog box closes and the initialization or a post-sort initialization file opens in the Property Panel for you to define.

For more information about the available data file formats, see “[Data File Formats and Types](#)” on [page 10](#).

Note: If you choose to create a new initialization or post-sort initialization data file by selecting **Insert > Data File**, you must specify the application mode from the **Application mode** list in the **New Data File** dialog box.

2.5.2 Defining the Initialization or Post-sort Initialization File Format

When you define the format of an initialization or a post-sort initialization file, you define the layout of the data and the format of the characters in the data source. You also define the data so that you can correctly map the data file in the Edit Panel.

To define the initialization or a post-sort initialization file format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.

3. To set up format-specific formatting, do one of the following, depending on the format you selected:

| If you selected this format | Do this |
|-----------------------------|--|
| Columnar | Continue to step 4. |
| Delimited | <p>a. In the Delimiter box, enter the character used to separate data.</p> <p>Note: For best results, OpenText recommends that you use delimiters from the standard ASCII printable range (32–126) for SBCS data files.</p> <p>b. If you want the delimiter to appear in hexadecimal code, select the Show hex check box.</p> <p>c. If the delimited data contains quoted fields, you must do the following:</p> <ol style="list-style-type: none">Select the Fields may be quoted with check box.In the adjacent box, enter the character used to surround data information. The engine ignores this character when it processes the data. |
| JSON | Complete the steps described in "Setting Up JSON Formatted Data Files" on page 253 . |
| ODBC | Complete the steps described in "Setting Up Data Files to Extract Data from an ODBC Data Source" on page 318 . |
| PDF Form | Complete the steps described in "Setting Up Data Files to Pre-Fill and Mine PDF Forms" on page 287 . |
| Web Service (RESTful) | Complete the steps described in "Setting Up Data Files to Communicate with a Web Service" on page 272 . |
| XML | Complete the steps described in "Setting Up XML Formatted Data Files" on page 178 . |

4. Go to the **Advanced** tab.

5. To define the character format of the data source, do one of the following, depending on the type of application in which you are working:

| For this type of application | Do this |
|------------------------------|--|
| SBCS | From the Character set list, select one of the following: <ul style="list-style-type: none">• Native—The engine determines the format at the data source automatically.• ASCII—The data source contains data in ASCII.• EBCDIC—The data source contains data in EBCDIC or it contains z/OS-specific formats such as packed decimal or zoned. If you are using EBCDIC files in an ASCII environment, the files must have a CR/LF (x0d0a) at the end of each record.• UTF-8—The data source is an XML or JSON file that uses UTF-8 encoding. This option is available only if you have licensed the XML/JSON Input module. |
| DBCS | From the Encoding box, select the encoding of the characters in the data source. The options vary based on the encodings enabled on your system. |

6. If the data source originates from a platform different from the one being used to map the data file, select one of the following options from the **Binary integer byte order** list in order to define the order of the bytes in the data source:
 - **Native**—Use the original byte order.
 - **Big-endian**—Convert the file to big-endian order. The first time you run Design Manager, it automatically converts existing EBCDIC data files to big-endian order.
 - **Little-endian**—Convert the file to little-endian order.

Note: Exstream Design and Production does not support data files with packed binary data for DBCS applications.

7. From the **Edit** menu, select **Save**.

2.5.3 Defining the Records in an Initialization or a Post-sort Initialization File

When you define the records in a initialization or a post-sort initialization file, you define how the data file separates the records of one customer from another customer and how the data file separates one individual record from another record. You also define which records you want to view and which ones are ignored when producing output. You cannot create sub-indicators in an initialization or a post-sort initialization file.

To define records in an initialization or a post-sort initialization file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the method in which customer records are separated at the data source, do one of the following:

| To | Do this |
|---|--|
| Specify that the data map in each line is unique (Columnar and Delimited) | From the How data is identified in the data file list, select Each record in the file has separate data mappings . |
| Specify that each line begins with a record type indicator (Columnar) | <ol style="list-style-type: none">a. From the How data is identified in the data file list, select File has specified record type(s).b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that each line has the same mapping (Columnar and Delimited) | From the How data is identified in the data file list, select Each record in the file has the same mapping . |
| Specify that each line begins with a record type indicator (Delimited) | <ol style="list-style-type: none">a. From the How data is identified in the data file list, select File has specified record type(s).b. In the Delimited field index of the first record-type indicator (1-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |

4. Go to the **Advanced** tab.
5. To specify how the data file separates one record from the next record, select one of the following options from the **Record type** list:
 - **CR/LF or LF**—Records end with a carriage return (CR) and line feed (LF), or an LF only. If you select this option and have licensed the On Demand Delivery module, the engine skips bad data records based on CR/LFs or LFs. If the bad data does not contain a CR/LF or LF, the engine combines it with records from the next data file until it encounters a CR/LF or LF in the read buffer.
 - **Fixed Length**—Records are the same length. In the **Length** box, enter the length of the records.
 - **Blocked (prefixed w/ 2 byte big-endian, exclusive)**—Records are preceded by a two-byte big-endian exclusive record length, which indicates the length of the record. This option is applicable only in the z/OS environment.
 - **CR/LF only**—Records end with a CR and an LF.
 - **MVS FTP block mode (prefixed w/ x80 + 2 bytes)**—Records are automatically

reblocked for FTP. This option is applicable only in the z/OS environment.

- **Blocked (prefixed w/ 4 byte big-endian, inclusive)**—Records are preceded by a four-byte big-endian inclusive record (two bytes of length followed by two null bytes). This option is applicable only in the z/OS environment.
6. To limit the number of records visible in the Edit Panel (for memory constraints or visual preference, for example) enter the number of records in the **Limit datamapping records** box. The default value is 1000. Enter 0 to specify that there are no constraints.
 7. To set initial data without using an initialization file, enter the number of records the headers occupy in the file in the **Header** box. When the engine runs, information in the header records (including the validation record) is not produced as output. During mapping, make sure the variables mapped to the header records have a setting of **Never reset** in the **Reset time** list.

For more information about variable properties, see “[Setting Up Variables](#)” on page 59.

Tip: If you are unsure of the exact number of header records in the data file, you can set the number higher than the actual number of header records. When the engine encounters a record specifying a customer, it stops reading header records.

8. From the **Edit** menu, select **Save**.

2.5.4 Selecting the Initialization or Post-sort Initialization File Data Source

When you select the data source for an initialization or a post-sort initialization file, you define one data source for mapping and testing and another data source for production. You map the test data source. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. The test data source is not the data source the engine uses to produce applications, however. During production, the engine uses the data maps, along with the production data source, to read and write customer data. The test data source and the production data source share many of the same properties to make sure your tests are as accurate as possible.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

To select the initialization or post-sort initialization file data source:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Advanced** tab.
3. To assign a connector that points to a routine that transforms incoming data before it reaches the engine, do one or both of the following steps based on when you want to use the connector.

| To | Do this |
|-------------------------------------|--|
| Use the connector during testing | From the Test record transform box, select an existing connector object from the Library. |
| Use the connector during production | From the Production record transform box, select an existing connector object from the Library. |

4. If you want the engine to verify that it is reading the correct source data, you must do the following:
 - a. Make sure the source data has a character string at the beginning that the engine can compare with the string you enter in the **Validation Record** box.
 - b. In the **Validation Record** box, enter the text that must match the string in the data source. The engine reads only the first 16 characters of the string.
5. In the **Rule** box, enter a rule to include or exclude the file during an engine run.
6. On the **Test Data Source** tab, you must do one of the following to specify the location of the data source to use for testing and mapping:

| To | Do this |
|--|--|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use for data mapping box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use for data mapping box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

7. On the **Production Data Source** tab, you must do one of the following to specify the location of the data source to use during production:

| To | Do this |
|--|--|
| Use data from an external file | <ol style="list-style-type: none">From the Type list, select File.In the File to use for data mapping box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">From the Type list, select Connector.In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

8. From the **Edit** menu, select **Save**.

2.6 Creating a Reference Data File

Reference files are optional files that let the engine access customer information not included in a customer driver file. A reference file lets you retrieve a supplemental set of customer data based on one or more key variables included in the customer driver file. The engine uses these keys to locate relevant data in the reference file. There is no limit to how much information you can tie to key variables or how many key variables you can use. If there are multiple key variables, all the information the engine reads after the first key is associated with that key. The engine associates all subsequent information with the first key until it finds a second key variable, and so on. This process continues until the engine encounters the end of the file. If your reference file has multiple keys, you must also create an auxiliary layout file in order to identify where the components of the key are located.

For more information about customer driver files, see “[Creating a Customer Driver Data File](#)” on [page 21](#).

For more information about creating an auxiliary layout file, see “[Creating an Auxiliary Layout Data File](#)” on [page 18](#).

An application can contain more than one reference file, and reference files can call other reference files. Since the engine processes reference files in the order in which they appear under the application in the Library, you must arrange reference files in the order that the data becomes available. For example, any reference file that refers to another reference file must occur after that reference file in the application.

To create a reference file, you must complete the following tasks:

1. [“Creating a Reference File Object” below](#)
2. [“Defining the Reference File Format” on the next page](#)
3. [“Defining the Records in a Reference File” on page 45](#)
4. [“Selecting the Reference File Data Source” on page 47](#)

2.6.1 Creating a Reference File Object

1. To select an application mode, complete one of the following tasks:

| To | Do this |
|--|--|
| Create a reference data file in SBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a reference data file in DBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a reference data file in SBCS/DBCS mode | <ol style="list-style-type: none">a. In Design Manager, in the Library, expand the Data Files heading.b. In the Library, right-click the type of data file you wish to create. For example, select SBCS data file for a new SBCS data file. The New Data File dialog box opens. |

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Reference file**.
4. To specify the format of the data source, select one of the following options from the **File format** list:
 - **Columnar**
 - **Delimited**
 - **JSON**
 - **Live**

- **ODBC**
 - **PDF Form**
 - **Print**
 - **XML**
5. To specify the source of the data, select one of the following options from the **File source type** list:
- **File**
 - **Web Service (RESTful)**
 - **Web Service (SOAP)**

Note: The Web Service (SOAP) source is not available for the JSON file format.

6. Click **Finish**.

The **New Data File** dialog box closes and the reference file opens in the Property Panel for you to define.

For more information about the available data file formats, see “[Data File Formats and Types](#)” on [page 10](#).

Note: If you choose to create a new reference data file by selecting **Insert > Data File**, you must specify the application mode from the **Application mode** list in the **New Data File** dialog box.

2.6.2 Defining the Reference File Format

When you define the format of a reference file, you define the layout of the data and the format of the characters in the data source. You define how and when the engine accesses the data in the reference file. You also define the data so that you can correctly map the data file in the Edit Panel.

To define the reference file format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the method in which the reference file is called from the customer driver file, you must do one of the following:

| To | Do this |
|---|--|
| Use one key variable | <ul style="list-style-type: none"> a. From the Reference key layout list, select Single variable. b. In the Key box, select the variable to use. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: The variable must appear in both the customer driver file and the reference file. It must also be less than 4MB. </div> |
| Use an auxiliary layout file to define multiple key variables | <p>From the Reference key layout list, select the name of the auxiliary layout file to use.</p> <p>For more information about auxiliary layout files, see “Creating an Auxiliary Layout Data File” on page 18.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: This option is not available for the JSON file format. </div> |

4. To define how the engine stores and accesses data from a reference file, select one of the following options from the **Access** list:

| Option | Description |
|--------------------|---|
| Memory | <p>The engine reads the entire reference file and stores all of the data in memory.</p> <p>The Memory option offers the fastest processing but uses the most memory. With small reference files, the Memory option yields the best results. Do not use this option if you want the engine to process section data or recipient data in the reference file.</p> |
| Disk seek | <p>The engine reads the entire reference file, keeps the reference keys in memory, and seeks the information as it encounters the keys.</p> <p>The Disk seek option is slower than the Memory option, but it uses less memory. Use this option for large reference files and to enhance performance time on ASCII-based processing systems.</p> |
| Create VSAM | <p>The engine creates a temporary ESDS (Entry Sequence Data Set) Virtual Storage Access Method (VSAM) file that serves as the reference file. The engine stores the keys in memory and seeks and reads the VSAM file for each reference.</p> <p>The Create VSAM option offers faster retrieval of data during lookup operations when the engine runs on z/OS. This option is applicable only in the z/OS and OS/390 environments.</p> <p>If you select this option, you must enter a production file name in the Temporary VSAM file in MVS JCL box. You must also specify both the temporary file name and the production file name of the reference file in the JCL. Do not use this option if you want the engine to process section data or recipient data in the reference file.</p> |
| Keyed VSAM | <p>The engine uses an existing KSDS (Keyed Sequence Data Set) VSAM file to identify a reference file whose data has already been stored in a VSAM file. The key can be either numeric or string.</p> <p>The Keyed VSAM option causes the engine to access information as it is needed so no keys are stored in memory. This option is applicable only in the z/OS and OS/390 environments.</p> |

| Option | Description |
|--------------------------|---|
| User routine | <p>The engine uses a custom routine that you have created.</p> <p>This option is available only if you have licensed the Dynamic Data Access module. Do not use this option if you want the engine to process section data or recipient data in the reference file.</p> |
| Driver-ordered, required | <p>The engine reads customer data in the customer driver file, looks for corresponding key variable values in the reference file, and then returns to the customer driver file for the next customer. The engine does not store keys in memory so the customer driver file and reference file values must be in the same order. If the reference file does not have a match at the same customer position, the engine reads the remainder of the reference file until it finds a match. The engine requires the specified key to be in the reference file. If the engine reaches the end of the reference file without a match, the run stops and you receive a severe error message.</p> <p>Select the Driver-ordered, required option if the customer driver file exactly matches reference files or if the customer driver file contains fewer customers than the reference file. Also use this option when the reference file contains extra keys that are not in the customer driver file (but the order is the same).</p> <p>This option is applicable only on columnar, delimited, JSON, and XML files. It does not support multiple customer driver files.</p> |
| Driver-ordered, optional | <p>The engine reads customer data in the customer driver file, looks for corresponding key variable values in the reference file, and then returns to the customer driver file for the next customer. The engine does not store keys in memory so the customer driver file and reference file values must be in the same order. If the reference file does not have a 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. If the engine finds a value in the reference file that does not have a corresponding match in the customer driver file, then all subsequent reads to the reference file fail.</p> <p>Select the Driver-ordered, optional option when the reference file is a subset of the customer driver file key values.</p> <p>This option is applicable only on columnar, delimited, JSON, and XML files. It does not support multiple customer driver files.</p> |

5. If you selected **Disk seek, Keyed VSAM, Driver-ordered, required, or Driver-ordered, optional** from the **Access** list, and you want the engine to recognize and read section data in the reference file, select the **Enable section data processing** check box.
6. If you selected **Disk seek, Keyed VSAM, Driver-ordered, required, or Driver-ordered, optional** from the **Access** list, and you want the engine to recognize and read recipient data records in the reference file and automatically create recipient copies from the data during processing, select the **Enable recipient copy processing** check box.

For more information about recipient data and automatically creating recipient copies, see *Designing Customer Communications* in the Exstream Design and Production documentation.

7. To set up format-specific formatting, do one of the following, depending on the format you selected:

| If you selected this format | Do this |
|-----------------------------|---|
| Columnar | Continue to step 8. |
| Delimited | <p>a. In the Delimiter box, enter the character used to separate data.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: For best results, OpenText recommends that you use delimiters from the standard ASCII printable range (32–126) for SBCS data files.</p></div> <p>b. If you want the delimiter to appear in hexadecimal code, select the Show hex check box.</p> <p>c. If the delimited data contains quoted fields, you must do the following:</p> <ol style="list-style-type: none">Select the Fields may be quoted with check box.In the adjacent box, enter the character used to surround data information. The engine ignores this character when it processes the data. |
| JSON | Complete the steps described in "Setting Up JSON Formatted Data Files" on page 253 . |
| Live | Complete the steps described in <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation. |
| ODBC | Complete the steps described in "Setting Up Data Files to Extract Data from an ODBC Data Source" on page 318 . |
| PDF Form | Complete the steps described in "Setting Up Data Files to Pre-Fill and Mine PDF Forms" on page 287 . |
| Print | Complete the steps described in "Setting Up Data Files to Mine Data from a Print File" on page 296 . |
| Web Service (RESTful) | Complete the steps described in "Setting Up Data Files to Communicate with a Web Service" on page 272 . |
| Web Service (SOAP) | Complete the steps described in "Setting Up Data Files to Communicate with a Web Service" on page 272 . |
| XML | Complete the steps described in "Setting Up XML Formatted Data Files" on page 178 . |

8. Go to the **Advanced** tab.

9. To define the character format of the data source, do one of the following, depending on the type of application in which you are working:

| For this type of application | Do this |
|------------------------------|--|
| SBCS | From the Character set list, select one of the following: <ul style="list-style-type: none">• Native—The engine determines the format at the data source automatically.• ASCII—The data source contains data in ASCII.• EBCDIC—The data source contains data in EBCDIC or it contains z/OS-specific formats such as packed decimal or zoned. If you are using EBCDIC files in an ASCII environment, the files must have a CR/LF (x0d0a) at the end of each record.• UTF-8—The data source is an XML or JSON file that uses UTF-8 encoding. This option is available only if you have licensed the XML/JSON Input module. |
| DBCS | From the Encoding box, select the encoding of the characters in the data source. The options vary based on the encodings enabled on your system. |

10. If the data source originates from a platform different from the one being used to map the data file, select one of the following options from the **Binary integer byte order** list in order to define the order of the bytes in the data source:
- **Native**—Use the original byte order.
 - **Big-endian**—Convert the file to big-endian order. The first time you run Design Manager, it automatically converts existing EBCDIC data files to big-endian order.
 - **Little-endian**—Convert the file to little-endian order.

Note: Exstream Design and Production does not support data files with packed binary data for DBCS applications.

11. To specify when the engine reads data from the reference file, select one of the following options from the **IO time** list:

| Option | Description |
|--|---|
| After initialization files read | The engine reads the data records in the reference file after it reads the initialization files. This option lets you summarize initialization information for reports before the production run affects it. This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered. |

| Option | Description |
|---|---|
| After initial customer data | <p>The engine reads the data records in the reference file immediately after it reads the customer's basic information. This option lets you summarize customer information for reports.</p> <p>Do not use the After initial customer data option when both the customer driver and reference files contain section data. With this option, the section data in the reference file comes before section data in the customer driver file. If you want the engine to read the customer driver file before the reference file, use the After named data section option.</p> |
| After each data section | <p>The engine reads the data records in the reference file after it reads each section of data in the customer driver file. This option is useful for section summaries, for chart and table data, and reports.</p> <p>This option is not supported on reference files that are driver-ordered.</p> |
| After customer data and each section | <p>The engine reads the data records in the reference file after the customer data and each section in the customer driver file, so the customer data can be included on a report followed by section data for section summaries.</p> |
| After named data section | <p>The engine reads the data records in the reference file after it reads a specified section of data in the customer driver file. In the Section box, enter the name of the section.</p> <p>This option is not supported on reference files that are driver-ordered.</p> |
| Start of queue and queue break | <p>The engine reads data records in the reference file when the output queue opens and at the top of each break file.</p> <p>For more information about output queues and break files, see <i>Creating Output</i> in the Exstream Design and Production documentation.</p> <p>This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered.</p> |
| Queue break and end of queue | <p>The engine reads the data records in the reference file after each break specified in Design Manager and after each output queue is complete. The engine creates reports on the file breaks and the end of output queue break.</p> <p>For more information about output queues and break files, see <i>Creating Output</i> in the Exstream Design and Production documentation.</p> <p>This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered.</p> |
| When each page is selected | <p>The engine reads the data records in the reference file after each page is accessed in a document. This option lets you summarize the information by page for reports.</p> <p>This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered.</p> |
| At the end of each customer | <p>The engine reads the data records in the reference file after each customer is complete. Each customer's information can be summarized for reports, including the current production run. The data is read before any campaign rules are processed.</p> <p>For more information about rules, see <i>Using Logic to Drive an Application</i> in the Exstream Design and Production documentation.</p> |

| Option | Description |
|--|--|
| Completion of all customers | The engine reads the data records in the reference file at the end of processing. This option creates a single output record upon completion of processing. It ensures that all of the variables are processed and written to the report in case the first records are shorter and contain less data than later records. This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered. |
| When TRIGGER called from rule/formula | The Trigger function is a special function that lets you specify your own timing for events. You can specify the exact time when you want the engine to read data. For more information about the Trigger function , see <i>Using Logic to Drive an Application</i> in the Exstream Design and Production documentation. This option is not supported on reference files with section data or recipient records, or reference files that are driver-ordered. |

12. From the **Edit** menu, select **Save**.

2.6.3 Defining the Records in a Reference File

When you define the records in a customer driver file, you define how the data file separates the records of one customer from another customer and how the data file separates one individual record from another record. You also define which records you want to view and which ones are ignored when producing output.

To define records in a reference file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the method in which customer records are separated at the data source, do one of the following:

| To | Do this |
|---|--|
| Specify that all customers have the same number of records (Columnar and Delimited) | <ol style="list-style-type: none">From the How to identify customers in the data file list, select Each customer has the same number of records.In the Number of records per customer box, enter the number of records per customer. |

| To | Do this |
|--|---|
| Specify that the file uses a single record type to indicate new customers (Columnar) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A new customer occurs on specified record type(s).b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that a new customer starts each time the record type indicator changes (Columnar) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A customer ID is on every record.b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that the file uses a single record type to indicate new customers (Delimited) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A new customer occurs on specified record type(s).b. In the Delimited field index of the first record-type indicator (1-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |
| Specify that a new customer starts each time the record type indicator changes (Delimited) | <ol style="list-style-type: none">a. From the How to identify customers in the data file list, select A customer ID is on every record.b. In the Delimited field index of the first record-type indicator (1-based) box, specify the beginning location of the first record type indicator.c. In the Size of the indicator box, specify the length of the record type indicator. |

4. If you are using indicators, and your data file has sub-indicators, you must do the following:
 - a. Select the **Multiple indicators** check box.
The **Edit Record Indicators** dialog box opens.
 - b. In the **Start** box, enter the start location.
 - c. In the **Length** box, enter the length of the sub-indicator.
 - d. Click **+**.
 - e. Click **OK**.
5. Go to the **Advanced** tab.
6. To specify how the data file separates one record from the next record, select one of the following options from the **Record type** list:
 - **CR/LF or LF**—Records end with a carriage return (CR) and line feed (LF), or an LF only. If you select this option and have licensed the On Demand Delivery module, the

engine skips bad data records based on CR/LFs or LFs. If the bad data does not contain a CR/LF or LF, the engine combines it with records from the next data file until it encounters a CR/LF or LF in the read buffer.

- **Fixed Length**—Records are the same length. In the **Length** box, enter the length of the records.
 - **Blocked (prefixed w/ 2 byte big-endian, exclusive)**—Records are preceded by a two-byte big-endian exclusive record length, which indicates the length of the record. This option is applicable only in the z/OS environment.
 - **CR/LF only**—Records end with a CR and an LF.
 - **MVS FTP block mode (prefixed w/ x80 + 2 bytes)**—Records are automatically reblocked for FTP. This option is applicable only in the z/OS environment.
 - **Blocked (prefixed w/ 4 byte big-endian, inclusive)**—Records are preceded by a four-byte big-endian inclusive record (two bytes of length followed by two null bytes). This option is applicable only in the z/OS environment.
7. To limit the number of records visible in the Edit Panel (for memory constraints or visual preference, for example) enter the number of records in the **Limit datamapping records** box. The default value is 1000. Enter 0 to specify that there are no constraints.
 8. To set initial data without using an initialization file, enter the number of records the headers occupy in the file in the **Header** box. When the engine runs, information in the header records (including the validation record) is not produced as output. During mapping, make sure the variables mapped to the header records have a setting of **Never reset** in the **Reset time** list.
- For more information about variable properties, see “[Setting Up Variables](#)” on page 59.
- Tip:** If you are unsure of the exact number of header records in the data file, you can set the number higher than the actual number of header records. When the engine encounters a record specifying a customer, it stops reading header records.
9. From the **Edit** menu, select **Save**.

2.6.4 Selecting the Reference File Data Source

When you select the data source for a reference file, you define one data source for mapping and testing and another data source for production. You map the test data source. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. The test data source is not the data source the engine uses to produce applications, however. During production, the engine uses the data maps, along with the production data source, to read and write customer data. The test data source and the production data source share many of the same properties to make sure your tests are as accurate as possible.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

To select the reference file data source:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Advanced** tab.
3. To assign a connector that points to a routine that transforms incoming data before it reaches the engine, do one or both of the following steps based on when you want to use the connector.

| To | Do this |
|-------------------------------------|--|
| Use the connector during testing | From the Test record transform box, select an existing connector object from the Library. |
| Use the connector during production | From the Production record transform box, select an existing connector object from the Library. |

4. If you want the engine to verify that it is reading the correct source data, you must do the following:
 - a. Make sure the source data has a character string at the beginning that the engine can compare with the string you enter in the **Validation Record** box.
 - b. In the **Validation Record** box, enter the text that must match the string in the data source. The engine reads only the first 16 characters of the string.
5. If information for a customer can change during processing (if you are using a connector to connect to the data source, for example) and you want to update the information in the data source, select the **File can be updated** check box.

Note: Use the Trigger function to control the timing of the update operation.

6. In the **Rule** box, enter a rule to include or exclude the file during an engine run. If the engine excludes the data file because of the rule, then the engine does not open the file or add it to internal file lists; instead, you receive an informational message.
7. From the **Rule run time** list, select one of the following options to specify when to process the rule:
 - **Once at startup**—The engine processes the rule only once, before the data file is opened.
 - **Before every operation**—The engine processes the rule before each lookup.

8. On the **Test Data Source** tab, you must do one of the following to specify the location of the data source to use for testing and mapping:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">From the Type list, select File.In the File to use for data mapping box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">From the Type list, select Connector.In the File to use for data mapping box, enter the file path to the external file used to provide data.In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

9. On the **Production Data Source** tab, you must do one of the following to specify the location of the data source to use during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">From the Type list, select File.In the File to use for data mapping box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. |
| | <p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">From the Type list, select Connector.In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

10. From the **Edit** menu, select **Save**.

2.7 Creating a Report or a Post-sort Report File

Report files are optional files that tell the engine how to write data. You can customize the file to report any information about the production run. When you run the engine, it generates the custom report in the location you specify with the file name you specify. The report file object must follow all other data files in the application. A post-sort report file serves the same function as a report file. However, the engine writes them only during the second pass of two-step processing.

Often, you create report files when you want to create audit or quality control files. You can create reports at the application-level or the output queue-level. If you create an application-level report, you receive a single report for the entire application. For example, you can track information about the total number of pages created during production. Output queue-level reports generate output queue-specific information rather than global information about an application. For example, you can track information about how many customers are processed through the output queue or how many documents are sent through the output queue.

For more information about creating output queue-level reports, see *Creating Output* in the Exstream Design and Production documentation.

If a record in a report file or a post-sort report file contains data areas with growing arrays, the engine replicates the record as needed in order to accommodate the elements. If the array has zero elements, the engine produces no records. If you map a report or a post-sort report and do not map some text or spaces, the text or spaces appear in the report unaltered, which is useful for titles and separators. Most commonly, the sample file uses the variable names as they appear in the customer driver file used to drive the application.

To create a report or a post-sort report file, you must complete the following tasks:

1. [“Creating a Report or Post-sort Report File Object” on the next page](#)
2. [“Defining the Report or Post-sort Report File Format” on page 52](#)
3. [“Defining the Records in a Report or Post-sort Report File” on page 55](#)
4. [“Selecting the Report or Post-sort Report File Template and Result Location” on page 56](#)

2.7.1 Creating a Report or Post-sort Report File Object

1. To select an application mode, complete one of the following tasks:

| To | Do this |
|---|--|
| Create a report or post-sort report data file in SBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a report or post-sort report data file in DBCS mode | In Design Manager, in the Library, right-click the Data Files heading and select New Data File . The New Data File dialog box opens. |
| Create a report or post-sort report data file in SBCS/DBCS mode | <ol style="list-style-type: none">In Design Manager, in the Library, expand the Data Files heading.In the Library, right-click the type of data file you wish to create. For example, select SBCS data file for a new SBCS data file. The New Data File dialog box opens. |

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select one of the following:
 - **Report file**
 - **Post-sort report file**
4. To specify the format of the data source, select one of the following options from the **File format** list:
 - **Columnar**
 - **Delimited**
 - **JSON**
 - **ODBC**
 - **PDF Form**
 - **XML**

Note: The JSON format is not available for post-sort report files.

5. To specify the source of the data, select one of the following options from the **File source type** list:
 - **File**
 - **Web Service (RESTful)**
 - **Web Service (SOAP)**

Note: The Web Service (SOAP) source is not available for the JSON file format.

6. Click **Finish**.

The **New Data File** dialog box closes and the report or the post-sort report opens in the Property Panel for you to define.

For more information about the available data file formats, see “[Data File Formats and Types](#)” on page 10.

Note: If you choose to create a new report or post-sort report data file by selecting **Insert > Data File**, you must specify the application mode from the **Application mode** list in the **New Data File** dialog box.

2.7.2 Defining the Report or Post-sort Report File Format

When you define the format of a report or post-sort report file, you define the layout of the data and the format of the characters in the sample data. You also define the sample data so that you can correctly map the data file in the Edit Panel.

To define the report or post-sort report file format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To set up format-specific formatting, do one of the following, depending on the format you selected:

| If you selected this format | Do this |
|-----------------------------|---------------------|
| Columnar | Continue to step 4. |

| If you selected this format | Do this |
|-----------------------------|--|
| Delimited | <p>a. In the Delimiter box, enter the character used to separate data.</p> <p>Note: For best results, OpenText recommends that you use delimiters from the standard ASCII printable range (32–126) for SBCS data files.</p> <p>b. If you want the delimiter to appear in hexadecimal code, select the Show hex check box.</p> <p>c. If the delimited data contains quoted fields, you must do the following:</p> <ol style="list-style-type: none">Select the Fields may be quoted with check box.In the adjacent box, enter the character used to surround data information. The engine ignores this character when it processes the data. |
| JSON | Complete the steps described in "Setting Up JSON Formatted Data Files" on page 253 . |
| ODBC | Complete the steps described in "Setting Up Data Files to Extract Data from an ODBC Data Source" on page 318 . |
| PDF Form | Complete the steps described in "Setting Up Data Files to Pre-Fill and Mine PDF Forms" on page 287 . |
| Web Service (RESTful) | Complete the steps described in "Setting Up Data Files to Communicate with a Web Service" on page 272 . |
| Web Service (SOAP) | Complete the steps described in "Setting Up Data Files to Communicate with a Web Service" on page 272 . |
| XML | Complete the steps described in "Setting Up XML Formatted Data Files" on page 178 . |

4. Go to the **Advanced** tab.
5. To define the character format of the data source, do one of the following, depending on the type of application in which you are working:

| For this type of application | Do this |
|------------------------------|---|
| SBCS | <p>From the Character set list, select one of the following:</p> <ul style="list-style-type: none">• Native—The engine determines the format at the data source automatically.• ASCII—The data source contains data in ASCII.• EBCDIC—The data source contains data in EBCDIC or it contains z/OS-specific formats such as packed decimal or zoned. If you are using EBCDIC files in an ASCII environment, the files must have a CR/LF (x0d0a) at the end of each record.• UTF-8—The data source is an XML or JSON file that uses UTF-8 encoding. This option is available only if you have licensed the XML/JSON Input module. |

| For this type of application | Do this |
|------------------------------|---|
| DBCS | From the Encoding box, select the encoding of the characters in the data source. The options vary based on the encodings enabled on your system. |

6. If the data source originates from a platform different from the one being used to map the data file, select one of the following options from the **Binary integer byte order** list in order to define the order of the bytes in the data source:
 - **Native**—Use the original byte order.
 - **Big-endian**—Convert the file to big-endian order. The first time you run Design Manager, it automatically converts existing EBCDIC data files to big-endian order.
 - **Little-endian**—Convert the file to little-endian order.

Note: Exstream Design and Production does not support data files with packed binary data for DBCS applications.

7. To specify when the engine writes data to the file, select one of the following options from the **IO time** list:

| Option | Description |
|--|--|
| After initialization files read | The engine writes the data records after it reads the initialization files. This option lets you summarize initialization information for reports before the production run affects it. <p>Caution: Use After initialization files read only when your application contains reference files.</p> |
| After initial customer data | The engine writes the data records immediately after it reads the customer's basic information. This option lets you summarize customer information for reports. |
| After each data section | The engine writes data records after it reads each section of data. This option is useful for section summaries, for chart and table data, and reports. |
| After named data section | The engine writes data records after it reads a specified section of data. This option enables flexibility for section headers in reports. In the Section box, enter the name of the section. |
| 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 | The engine writes data records after each break specified in Design Manager and after each queue is complete. The engine creates reports on the file breaks and end of queue break. |
| When each page is selected | The engine reads the reference file after each page is accessed in a document. This option lets you summarize the information by page for reports. |

| Option | Description |
|--|---|
| At the end of customer, before campaigns | The engine writes data records after each customer is complete. Each customer's information can be summarized for reports, including the current production run. The data is read before any campaign rules are processed. |
| Completion of all customers | The engine writes data records after it finishes processing. This option creates a single output record upon completion of processing. It ensures that all of the variables are processed and written to the report in case the first records are shorter and contain less data than later records. |
| As determined by record properties | The engine writes data records using the properties set on the data record. |
| When TRIGGER called from rule/formula | The Trigger function is a special function that lets you specify your own timing for events. You can specify the exact time when you want the engine to write data. For more information about the Trigger function, see <i>Using Logic to Drive an Application</i> in the Exstream Design and Production documentation. |
| Based on standard customer/section breaks | The engine writes data records based on the customer and section breaks found in the data file. |

8. From the **Edit** menu, select **Save**.

2.7.3 Defining the Records in a Report or Post-sort Report File

When you define the records in a report or a post-sort report file, you define the formatting of individual records in the data file. You also define which records you want to view.

To define records in a report or post-sort report file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Advanced** tab.
3. To specify how the data file separates one record from the next record, select one of the following options from the **Record type** list:
 - **CR/LF or LF**—Records end with a carriage return (CR) and line feed (LF), or an LF only. If you select this option and have licensed the On Demand Delivery module, the engine skips bad data records based on CR/LFs or LFs. If the bad data does not contain a CR/LF or LF, the engine combines it with records from the next data file until it encounters a CR/LF or LF in the read buffer.
 - **Fixed Length**—Records are the same length. In the **Length** box, enter the length of the records.
 - **Blocked (prefixed w/ 2 byte big-endian, exclusive)**—Records are preceded by a

two-byte big-endian exclusive record length, which indicates the length of the record.
This option is applicable only in the z/OS environment.

- **CR/LF only**—Records end with a CR and an LF.
 - **MVS FTP block mode (prefixed w/ x80 + 2 bytes)**—Records are automatically reblocked for FTP. This option is applicable only in the z/OS environment.
 - **Blocked (prefixed w/ 4 byte big-endian, inclusive)**—Records are preceded by a four-byte big-endian inclusive record (two bytes of length followed by two null bytes).
This option is applicable only in the z/OS environment.
4. To limit the number of records visible in the Edit Panel (for memory constraints or visual preference, for example) enter the number of records in the **Limit datamapping records** box. The default value is 1000. Enter 0 to specify that there are no constraints.
 5. From the **Edit** menu, select **Save**.

2.7.4 Selecting the Report or Post-sort Report File Template and Result Location

Report and post-sort report files require you to select the location of the template you use to map the report and the locations where the test and production versions of the report will be created. Since report files instruct the engine to create data rather than read data, you do not map the test data source. Instead, you map a separate data template. The test data location you specify is the location of where the report is created when you test an application.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

To select the report or post-sort report file template and result location:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. In the **Sample file to use to map report data files** box, enter the path to the report or post-sort report file template. Do not use the file name you enter in the box on the **Test Data Source** or **Production Data Source** tabs. If you do use the same file name, the output writes over the contents of the data template.
4. Go to the **Advanced** tab.
5. In the **Rule** box, enter a rule to include or exclude the file during an engine run. Although it is possible to place rules on report files on an output queue, these rules are not processed, and you receive no message.
6. To specify when to process the rule, select one of the following options from the **Rule run time** list:

- **Once at startup**—The engine processes the rule only once, before the data file is opened.
 - **Before every operation**—The engine processes the rule before each lookup.
7. On the **Test Data Source** tab, you must do one of the following to specify the location of the report created during testing:

| To | Do this |
|--|--|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use for data mapping box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use for data mapping box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

8. On the **Production Data Source** tab, you must do one of the following to specify location of the report created during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in production box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

9. From the **Edit** menu, select **Save**.

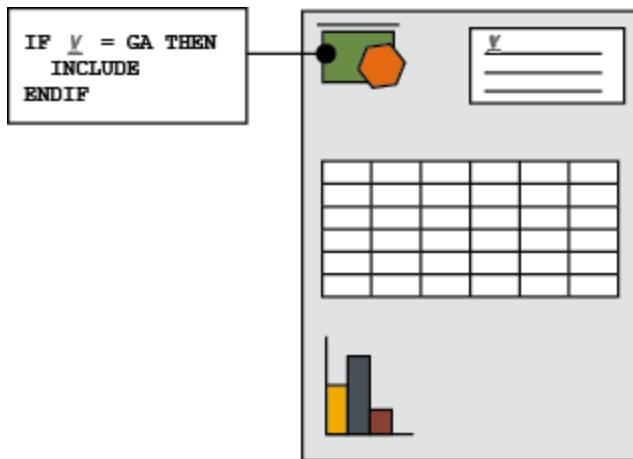
Chapter 3: Setting Up Variables

Variables hold the place for dynamic content. The content generally comes from three locations: data in a data file, information stored on the design or production system, or information generated during packaging or production. You can use the data from a data file during design to create output customized for each customer. You can use a variable directly on a page to personalize a design using customer data, or in a rule to include or exclude an object from a design or final output. The information stored on the design or production system is useful if you want to include information such as the date in a design or in a report. The information generated during packaging and production lets you create reports about the progress of a production run or send information used for high-volume processing to the production engine.

For more information about using variables in a design, see *Designing Customer Communications* in the Exstream Design and Production documentation.

For more information about using variables in a rule or formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Variables in a design



When you map a data file, you associate a variable with a data area. This process creates data maps that the engine uses to find data. The engine then uses the data to populate variables and customize documents. The variable mapped to the data area defines the type of data found in the area. During mapping, you also define how to separate different types of data from each other and the use and format of the data area. For example, suppose the data in the area is a date. When the engine populates the variable during production, the date in the specified area replaces the variable.

For more information about using variables during production, see *Switch Reference* in the Exstream Design and Production documentation.

This chapter discusses the following topics:

- “[Types of Variables](#)” below
- “[Creating Variables](#)” on page 64
- “[Formatting Variable Values in Output](#)” on page 81
- “[Testing Variables](#)” on page 99

3.1 Types of Variables

Unlike data files, which use a single feature to define the type, variable types are defined using a number of different traits. This multi-trait definition lets you customize your variable to represent your exact data needs. You can even use these traits to filter the variables you view in the Variable Palette. By filtering the variables, you make it easier to find just the one you want.

Variables can be categorized into different types using a combination of the following three traits:

- “[Variable Sources](#)” below
- “[Data Type Represented by a Variable](#)” on page 62
- “[Number of Values a Variable Can Represent](#)” on page 63

3.1.1 Variable Sources

All variables come from one of two sources: the user or the system. The variables you create yourself to specifically take data from a data file and then use the data in a design, in a report, or in a rule are called user-defined variables. User-defined variables can have any name as long as the name meets the conventions as defined by Exstream Design and Production. You can also store the variables in any folder.

For more information about the naming conventions, see “[Variable Naming Conventions](#)” on the next page.

System variables let you use system information, such as the information stored in the design environment or the information generated as part of the packaging process, to complete tasks such as monitoring and controlling an engine run, formatting designs, or organizing a customer mailing. Both the variable name and the information provided by the variable are predefined and you cannot change either.

System variables are always stored under the **Data Dictionary** heading under the **Exstream** root folder. You cannot delete a system variable from the root folder. If you want to use a system variable in another folder, you must create a clone. All system variables are always included with Exstream Design and Production and they are not module dependent. While you use some

system variables more often with a particular module, you are not required to license a module to use a system variable.

During packaging, Exstream Design and Production automatically adds the following four system variables to your package file, whether you use them or not in the design of the application: 'SYS_LanguageCustomer', 'SYS_LocaleCustomer', 'SYS_CustomerEffectiveDate', and 'SYS_CustInvalidDataLevel'. These system variables are used internally by Exstream Design and Production and refer to commonly used information such as the language of the customer being processed by the engine. They are included to ensure the information they represent is included in the system report that is available after an engine run, in an application report, and in any messages issued during the packaging process.

For more information about package files, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

Customizing How the Value of a System Variable is Calculated

Although you cannot change the type of information provided by a system variable, some system variables let you use a rule or formula to customize the method in which the values are calculated. For example, the 'SYS_LanguageCustomer' system variable requires you to create a rule to specify the language for a customer. You can then use the language definition to select the correct language-related object during production.

For more information about which system variables values can be customized, see ["System Variables" on page 103](#). The names of the system variables with values that can be customized are bold.

To customize how the value of a system variable is calculated:

1. In Design Manager, from the Library, drag the system variable to the Property Panel.
2. Go to the **Basic** tab of the variable's properties.
3. Verify that the **can be set** check box is selected. The check box is always inactive, meaning that you cannot select or clear it, but if the check box is selected, you can create a rule or formula to set the value of the system variable.
4. Go to the **Values** tab.
5. In the **Formula** area, create the rule or formula for the system variable.
6. From the **File** menu, select **Save**.

For more information about how to create a formula in a system variable, see ["Using a Formula to Compute Variable Values" on page 68](#).

Variable Naming Conventions

The names of system variables are predefined. They always begin with "SYS." The system variables designed for use with interactive and Web-based documents begin with "SYSLD" or

"SYSWEB." You cannot change the name of a system variable.

You can name user-defined variables according to your organization's naming conventions; however, they must comply with the following conventions:

- Variables names can contain only alphanumeric characters and underscores.
- Variable names must be fewer than 255 characters.
- Variable names can contain European accent characters; however, you cannot use Unicode characters.
- Variables names must contain at least one alpha character.
- Variable names cannot start with "x" followed by numbers (as in "X18") since they are interpreted as an integer hex constant in a rule, formula, or function.
- Variables names cannot begin with a number if you use the variable in a rule or formula where the variable is qualified with the folder where the variable is located.
- Variable names cannot include the name of a built-in command or function, such as Sort, or a Visual Basic command, like Endif.

Note: When working with variables, keep in mind that the SBCS engine cannot process variables whose names contain DBCS characters. Variables that are used in both SBCS and DBCS applications must have names that are made up of SBCS characters only.

For more information about built-in functions, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

3.1.2 Data Type Represented by a Variable

When you create a variable, you must select the type of data that the variable represents. The data type is preselected on system variables. This data type defines properties such as the available formatting of the variable when it is included in the output, how designers can use the variable in a design, and how the engine calculates the value of the variable. Some data types also let you customize the formatting in the output based on the customer language. For example, if you are creating a variable to represent the days of the week, the values can change based on the language you specified for a customer.

For more information about creating a variable, see ["Creating Variables" on page 64](#).

The following table lists the data types available:

Data types represented by variables

| Type | Icon | Data referenced |
|---------|---|---|
| Boolean |  | Boolean variables represent data that must be one of only two possible values. For example, the value of a Boolean variable is always true or false and can have no other values. |

Data types represented by variables, continued

| Type | Icon | Data referenced |
|----------------|------|--|
| Currency | \$ | Currency variables represent data that is a monetary amount. These variables can have up to 14 digits to the left of the decimal and four digits to the right of the decimal. The formatting can be language-dependent. |
| Date | BT | Date variables represent data that is a date, a time, or both. Formatting can be language-dependent. Note: The date 02/29 is a valid value for the date variable only when the associated year is a leap year. |
| Floating | .x | Floating variables represent data that is a numeric value that includes decimals. Floating variables are IEEE 754 single-precision values. Formatting can be language-dependent. |
| Formatted text | FT | Formatted text variables represent data that is formatted content that comes from an external repository or file source. For more information about formatted text variables, see <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation. |
| Integer | I | Integer variables represent data that is a whole number. The range of the possible field length of an integer variable is negative 2 billion to positive 2 billion. |
| Placeholder | PH | Placeholder variables represent data that is content from an external source and is accessed during an engine run. This variable type requires the Dynamic Content Import module. For more information about placeholder variables, see <i>Importing External Content</i> in the Exstream Design and Production documentation. |
| String | A | String variables represent data that contains text. The size limit of string variables in Exstream Design and Production is 64K. String variables can be language-dependent. |
| Tagged text | CT | Tagged text variables represent data that has tags specifying the formatting of imported text. In the default tag set, Design Manager uses angle brackets (<>) for tagged text, but you can use any character to define a tag set and denote tagged text. When using tag sets with XML input files, which denote elements using angle brackets, you must use different characters, such as square brackets ([]) to denote tagged text. For more information about tagged text variables, see <i>System Administration</i> in the Exstream Design and Production documentation. |

3.1.3 Number of Values a Variable Can Represent

The number of values a variable can represent varies. A variable can always represent a single value for each customer, or the variable can represent multiple values for each customer.

If a variable represents only one value per customer (for example, the customer name), it is called a scalar variable. If a variable represents multiple values per customer, it is called an array variable. If the number of values in an array is constant for each customer (for example, the number of months in a year), it is called a static array. If the number of values in an array

varies for each customer (for example, the number of transactions in a month), it is called a growing array. To use array variables, the system settings must be configured to allow static and growing arrays in the design.

For more information about configuring the system settings, see *System Administration* in the Exstream Design and Production documentation.

3.2 Creating Variables

The basic process of creating a variable is the same regardless of the type of variable you are creating. Creating a variable involves defining the method by which the engine calculates the value of the variable and when, or if, a designer can use the variable. You also specify the valid values of the variable to ensure the information the variable includes is correct. In addition, you specify the information about the variable to include when creating a report for troubleshooting purposes. Later, you can access variables you create, filter them, and insert them into designs from the Variable Palette in Design Manager or Designer.

For more information about inserting a variable in a design and filtering variables in the Variable Palette, see *Designing Customer Communications* in the Exstream Design and Production documentation.

To create a variable, you must complete the following tasks:

1. “[Creating a Variable Object](#)” below
2. “[Specifying the Variable Values](#)” on the next page
3. “[Selecting Variable Validation and Troubleshooting Methods](#)” on page 72

You can also complete the following optional tasks as needed:

- “[Tagging Variables with Metadata](#)” on page 74
- “[Creating Variables Automatically from XML or an XML Schema](#)” on page 74
- “[Populating a String Variable Using a String Lookup Table](#)” on page 78
- “[Adding Variables to an Application](#)” on page 79
- “[Specifying How Array Variable Elements are Displayed](#)” on page 80

3.2.1 Creating a Variable Object

1. In Design Manager, in the Library, right-click the **Data Dictionary** heading and select **New Variable**.

The **New Variable** dialog box opens.

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **Type** list, select the one of the following options:
 - **Boolean**
 - **Currency**
 - **Date**
 - **Floating**
 - **Formatted text**
 - **Integer**
 - **Placeholder**
 - **String**
 - **Tagged text**
4. If the variable can represent more than one value per customer, select the **Array** check box.
5. In the **Design sample** box, enter the text string you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples in DBCS applications.
6. Click **Finish**.

The **New Variable** dialog box closes and the variable opens in the Property Panel for you to define.

Tip: Entering a description and design sample when creating variables helps you identify them in reports, messages, and external programs.

3.2.2 Specifying the Variable Values

When you specify the value of a variable, you must define information such as the number of values the variable can represent, the method for generating the value, and in order to calculate the value correctly, when the variable is available. Keep in mind that the variable reset times and compute times affect its behavior when used with section data.

For more information about section data, see [“Section Data in Exstream Design and Production” on page 4](#).

During production, the engine populates the values of user-defined variables at different times. For example, you can create a variable that populates each time the engine encounters it in a design or you can create a variable that places page numbers after pages have been included

based on marketing messages or excluded because of rules. The values of system variables are always computed as needed by the engine during production.

Tip: If you experience errors when using variables on specific objects, or if variables do not function as expected, verify the timing of when the engine populates the variable.

To specify the variable value:

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Basic** tab.
3. From the **Data type** list, verify the selection. If you selected the wrong type, you can select a different option.
4. To specify the number of values for a customer or section of data, you must do one of the following:

| To | Do this |
|---|---|
| Use a scalar variable with a single value per customer | Clear the Array check box. |
| Use a growing array variable with a varying the number of values per customer | <ol style="list-style-type: none">Select the Array check box.From the list, select Grows. |
| Use a static array variable with a set number of values per customer | <ol style="list-style-type: none">Select the Array check box.From the list, select Static.In the adjacent box, enter the number of elements in the array. |

5. To specify the method for generating the values of the variable, you must complete one of the following tasks:

| To | Do this |
|--|---|
| Specify the variable values in the variable properties | Complete the steps in "Specifying the Variable Values in the Variable Properties" on the next page. |
| Use a formula to compute or create variable values | Complete the steps in "Using a Formula to Compute Variable Values" on page 68. |
| Read the variable values from a data file | Complete the steps in "Reading the Variable Values from a Data File" on page 70. |
| Create a cross-reference variable | Complete the steps in "Creating a Cross-Reference Variable" on page 70. |
| Create a variable that counts events | Complete the steps in "Creating a Variable That Counts Events" on page 70. |

6. To define when the variable is available for use, select one of the following options from the **Design use** list:
 - **None**—The variable is unavailable for use in rules or design objects and is not included in variable lists. Select this option when you want to prevent designers from using the variable, but you do not want to delete it. For example, you do not want to use a variable from a legacy application in new application, but, because the legacy application is still in production, you cannot delete the variable.
 - **Can only be used to personalize**—The variable can be used only for personalizing design objects directly in a design. It cannot be used in rules.
 - **Can only be used in rules**—The variable can be used only to create rules on design objects. It cannot be used directly in a design.
 - **Rules or personalization**—The variable can be used for both personalization and in rules. This option is the default setting.
 - **Selector**—The variable can be used to create selection objects in LiveEditor.

For more information about LiveEditor, see *Designing for LiveEditor* in the Exstream Design and Production documentation.

7. For all variable sources, except **Counter**, select one of the following options from the **Reset time** list to specify when the variable value resets to zero elements:
 - **Automatically**—Resets before each customer. If you map the variable to a section-based data file, the engine resets the value before it reads any sections.
 - **Before each customer**—Resets only before each customer is read, regardless of section-based data
 - **Never reset**—Resets only at the beginning of each engine run
 - **Named section**—Resets before the engine reads the section specified in the adjacent **Named section** box
 - **All sections**—Resets before all sections
8. From the **Edit** menu, select **Save**.

Specifying the Variable Values in the Variable Properties

1. From the **Source** list, select **User value**.
2. Go the **Values** tab.
3. If you want to use the variable with language layers and you want a separate initial value for each language, you must do the following:

- a. Select the **Set initial values for each language** check box.
- b. From the list, select the language for which you want to specify initial values. The languages available vary based on the languages in your database.
4. If you are defining a static array variable, in the **Array element** box, select the array element to define.
5. In the **Initial value** box, enter the initial value up to 64K characters. The initial value can be up to 32K for DBCS applications.

For more information about using language layers, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Using a Formula to Compute Variable Values

1. From the **Source** list, select **Formula**. Or, if you are defining a system variable, verify that the **can be set** check box is selected.
2. Go the **Values** tab.
3. In the **Formula** box, enter a rule or formula. If you use an external editing program to create logic, click **>>** above the **Formula** box to open the editor and enter the formula in the editor. You can right-click inside the **Formula** box to access a shortcut menu that lets you quickly access commonly used components.
4. To specify when you want the engine to compute the formula, select one of the following options from the **Compute time** list:
 - **As-needed**—Computes when the variable is referenced in other objects. Do not use **As-needed** for variables on late compose objects. If you select **As-needed**, the formula cannot contain references to itself.

Tip: For variables in schema model data files, it is a best practice to select a **Compute time of As-needed**. This option optimizes the way in which the engine accesses variable data in schema model data files, because for schema model data files, all of a customer's data is made available at the beginning of the processing of that customer. This data availability means that variables in schema model data files are best computed as they are encountered in the design.

For more information about schema model data files, see “[Creating Schema Model Data Files](#)” on page 215

- **All sections**—Computes for each section as it is encountered in the data
- **Completion**—Computes after processing the last customer
- **Customer**—Computes for each customer, before any section data

- **Initialization**—Computes before processing the first customer
 - **Named section**—Computes after the engine reads the named section specified in the adjacent box
 - **Data aggregation**—Computes after the data aggregation event
 - **Don't compute**—Does not compute the variable. Use this option if the variable is computed only in a post-sort processing run.
 - **Page**—Computes for each page
 - **Post-customer**—Computes for each customer, after all section data
5. If you have licensed the Output Sorting and Bundling module and you are creating a formula that computes during a post-sort processing run, select one of the following options from the **Post-sort compute time** list:
- **As-needed**—Computes when the variable is referenced in other objects
- Tip:** For variables in schema model data files, it is a best practice to select a **Post-sort compute time** of **As-needed**. This option optimizes the way in which the engine accesses variable data in schema model data files, because for schema model data files, all of a customer's data is made available at the beginning of the processing of that customer. This data availability means that variables in schema model data files are best computed as they are encountered in the design.

For more information about schema model data files, see "[Creating Schema Model Data Files](#)" on page 215
- **Page**—Computes for each page. If the **Compute time** option is set to **Page**, then the **Post-Sort compute time** option is set to **Page** and disabled.
 - **Post-customer**—Computes for each customer, after all section data
 - **Initialization**—Computes before processing the first customer
 - **Completion**—Computes after processing the last customer
 - **Don't compute**—Does not compute the variable
6. To define when to record information about each line of a formula as it is read by the engine, select one of the following options from the **Code trace** list:
- **None**—The formula code is not traced. This option is the default.
 - **Source Line**—Each line of the formula is written to the debug file as the rule is executed.
 - **Assignment**—Each line of the formula is written to the debug file as the rule is executed, along with the variables as they are used.

- **All Variables**—Each line of the formula is written to the debug file as the rule is executed, along with the values of variables as they are used.

For more information about how to create a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

For more information about the Output Sorting and Bundling module, see *Creating Output* in the Exstream Design and Production documentation.

For more information about debug files, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

Reading the Variable Values from a Data File

1. From the **Source** list, select **File only**.
2. If you are defining a scalar or a static array variable and want to set the initial values:
 - a. Go the **Values** tab.
 - b. In the **Array element** box, select the array element to define. The number varies based on the number of elements in the variable.
 - c. In the **Initial value** box, enter the initial value up to 64K characters. The initial value can be up to 32K for DBCS applications.

Creating a Cross-Reference Variable

1. From the **Data type** list, select **Integer**.
2. From the **Source** list, select **Crossref page number**.

For more information about cross-references, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Creating a Variable That Counts Events

Note: You cannot count XML node-based events in schema model data files using a counter variable.

To create a variable that counts events:

1. From the **Source** list, select **Counter**.
2. Make sure that the **Array** check box is cleared.
3. Click the **Values** tab.

4. In the **Initial value** box, enter the initial value from which the counter should increment, up to 64K characters. The initial value can be up to 32K for DBCS applications.
5. In the **Formula** box, you can also enter logic that computes the value by which the counter should increment. If you use an external editing program to create logic, click **>>** above the **Formula** box to open the editor. If you do not enter a formula, the counter increments by one.
6. To define when you want the counter to increment, select one of the following options from the **Compute time** list:
 - **All sections**—Increments as each section is encountered in the data
 - **Completion**—Increments after processing the last customer
 - **Customer**—Increments for each customer, before any section data
 - **Initialization**—Increments before processing the first customer
 - **Named section**—Increments after the engine reads the named section that is specified in the adjacent box
 - **Data aggregation**—Increments after the data aggregation event
 - **Don't compute**—Does not increment the counter. Use this option if the counter should increment only in a post-sort processing run.
 - **Page**—Increments for each page
 - **Post-customer**—Increments for each customer, after all of the section data
7. If you have licensed the Output Sorting and Bundling module, select one of the following options from the **Post-sort compute time** list:
 - **Page**—Increments for each page. If the **Compute time** option is set to **Page**, then the **Post-Sort compute time** option is set to **Page** and disabled.
 - **Post-customer**—Increments for each customer, after all of the section data
 - **Initialization**—Increments before processing the first customer
 - **Completion**—Increments after processing the last customer
 - **Don't compute**—Does not increment the counter

For more information about the Output Sorting and Bundling module, see *Creating Output* in the Exstream Design and Production documentation.

3.2.3 Selecting Variable Validation and Troubleshooting Methods

To make sure variables provide accurate results, you can validate the data as it is read. You can also gather information about the variable during processing to make troubleshooting later easier. Since you can use variables in multiple places and the values can come from multiple sources, you can select options to make sure the variable value is valid. When you validate the data, you ensure that formulas are calculated correctly and that the output includes the values you expect. You can also verify that the engine is populating the variables as expected by selecting which information to include in the debug file created at the end of an engine run.

For more information about debug files, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

To select variable validation and troubleshooting methods:

1. For all variable sources, except **Crossref page number**, drag the variable from the Variable Palette to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the method for automatically validating that the proper value has been set for the variable, you must do one of the following:

| To | Do this |
|---|---|
| Specify that the variable value is not validated when the value changes | From the Validation method list, select None . |
| Specify a range of valid values | <ol style="list-style-type: none">a. From the Validation method list, select Value range.b. In the Valid values boxes, enter the minimum and maximum values of the variable. |
| Specify a valid length range | <ol style="list-style-type: none">a. From the Data type list, select String.b. From the Validation method list, select Length range.c. In the Valid values boxes, enter the minimum and maximum lengths of the variable. |
| Specify a precise valid value | <ol style="list-style-type: none">a. From the Validation method list, select Equal to.b. In the first Valid values box, enter the exact value the variable value must match. |
| Specify a value that the variable value must be greater than | <ol style="list-style-type: none">a. From the Validation method list, select Greater than.b. In the first Valid values box, enter the value that the variable value must be greater than. |

| To | Do this |
|--|---|
| Specify a value that the variable value must be greater than or equal to | a. From the Validation method list, select Greater than or equal to . b. In the first Valid values box, enter the value that the variable value must match exactly or be greater than. |
| Specify a value that the variable value must be less than | a. From the Validation method list, select Less than . b. In the first Valid values box, enter the value that the variable value must be less than. |
| Specify a value that the variable value must be less than or equal to | a. From the Validation method list, select Less than or equal to . b. In the first Valid values box, enter the value that the variable value must match exactly or be less than. |
| Specify a precise value that is not valid | a. From the Validation method list, select Not equal to . b. In the first Valid values box, enter the exact value that the variable cannot match. |
| Specify a Boolean function that returns true if the value is valid | a. From the Validation method list, select Function . b. In the Validation function box, select a function. The function should refer to the data entered by the 'SYS_ValidationVariable' system variable. The variable is used to access the data to be validated within the function. If the function cannot be located or returns an error or a severe error, the validation fails. |

4. To specify the action the engine should perform when it finds an invalid value, select one of the following options from the **Action if invalid** list:
 - **Message, continue**—You receive a message and the engine continues processing.
 - **Message, set to default**—You receive a message and the engine resets the variable to its default value and continues processing.
 - **Message, stop**—You receive a message and the engine stops processing.
 - **Message, skip document**—The engine skips the document if the variable is not valid or is not found in the data.
5. To specify the variable tracking method and when the engine writes information to a debug file, select one of the following options from the **Watch level** list:
 - **None**—The engine disables watch level activities.
 - **Set**—The engine writes information when the variable is set.
 - **Changed**—The engine writes information each time the variable changes.
6. From the **Edit** menu, select **Save**.

3.2.4 Tagging Variables with Metadata

You can use the metadata feature in Exstream to tag variables after you create them. Metadata functions as a descriptive attribute by which you can later filter variables in the Variable Palette in Design Manager or Designer. For example, suppose you tag all variables that store contact information such as phone number and address information with a metadata object named "ContactInfo." When you map the contact information in a data file, you could easily find the variables you need by filtering the Variable Palette for the variables tagged with "ContactInfo."

You can also tag variables with multiple metadata objects. For example, you might tag variables that store customer address information with both "ContactInfo" and "AddressInfo." If you later filter variables for the "ContactInfo" metadata object, you will see the variables you tagged that store address information, as well as variables you tagged that store phone numbers and any other contact information. If you filter for "AddressInfo," you will see only the variables you tagged that store address information.

Before you can add metadata to a variable, you must first create the appropriate metadata objects.

For more information about creating metadata and using the metadata feature, see *Designing Customer Communications* in the Exstream Design and Production documentation.

To tag a variable with metadata:

1. In Design Manager, in the Library, expand the **Data Dictionary** heading.
2. Right-click the variable to which you want to add metadata and select **Metadata**.

The **Select Metadata** dialog box opens.

3. Select one or more check boxes next to the metadata you want to add to the object.
4. Click **OK**.

The **Select Metadata** dialog box closes.

3.2.5 Creating Variables Automatically from XML or an XML Schema

If you are using an XML data file or an XML schema in your design, you can create variables automatically using one of these files. For example, suppose you need to create variables to map to an XML data file and you do not yet have data, but you have a schema that defines the XML data. You can use the schema to automatically create the variables that you will eventually map to XML data. You can also use a schema included in a Web Service Description Language (WSDL) file.

When you create variables from XML or an XML schema, Design Manager selects the basic name, data type, and format features of a variable based on the information in the file. During the

automatic creation process, you can change variable names and data types as necessary, and you can ignore tags for which you do not want to create variables. Additionally, you can add metadata to variables during automatic creation. For example, you might add a metadata object that identifies a schema used to create the variables so that you can easily identify those variables later in the design process.

For more information about inserting a variable in a design and filtering variables in the Variable Palette, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Design Manager detects data types more accurately when using a schema instead of XML data, since a schema contains data type information. Keep in mind that schemas can define data types that do not match data types in Exstream Design and Production. In this situation, Design Manager substitutes a supported data type when creating the variables. If you use XML data to create variables automatically, Design Manager detects data types automatically, but you might need to make minor adjustments. For all automatically created variables, make sure that you verify all of the variable properties.

To create variables automatically from XML or an XML schema:

1. In Design Manager, in the Library, right-click the **Data Dictionary** heading and select **Create Variables from XML**.

The **Variables from XML** dialog box opens.
2. In the **Variable creation source file** box, enter the path and name, or the URL, of a source file, or click  to browse to a file. You can use an XML file, an XML schema, or a WSDL file including an XML schema.
3. From the **File type** list, select the type of file you are using to create variables:
 - **XML**—The file contains XML data.
 - **Schema**—The file contains an XML schema.
 - **WSDL**—The file describes a web service using WSDL and includes an XML schema.
4. From the **Encoding** box, select the encoding of the characters in the data source. The options vary based on the encodings enabled on your system.
5. If you selected **Schema** from the **File type** list, select from the **Select root element** list the root element defined by the schema that will be used as the root element in your XML data.
6. If you selected **WSDL** from the **File type** list, select from the **SOAP operation** list the SOAP operation that identifies the schema in the WSDL file.
7. If you want to use values of an attribute instead of tag names to identify the XML hierarchy, select the **Replace tag name with attribute value** check box, and then enter the attribute name to use in the **Attribute name** box.

For more information about using an attribute to identify XML hierarchy, see “[Using Attributes to Identify the XML Hierarchy](#)” on page 208.

8. If you want to add a metadata object to all variables created from the XML file or schema so that you can later filter the Variable Palette to see only those variables, complete the following steps:
 - a. Select the **Include metadata for variables** check box.
 - b. To select or create a metadata object, you must do one of the following:

| To | Do this |
|------------------------------------|--|
| Select an existing metadata object | Click  and select the metadata object to add to the new variables. |
| Create a new metadata object | <ol style="list-style-type: none">i. Click New. The New Metadata dialog box opens.ii. In the Name box, enter a name. In the Description box, enter a description (optional).iii. If you do not want a value associated with the metadata, click the No value check box.iv. If you did not select the No value check box and you want to assign a default value, enter the default text value you want to assign to the metadata in the Default value box, or click  and select a variable you want to assign as the default value. |

- c. If the metadata object you selected allows values (the **No value** check box is not selected), enter a value for the metadata in the **Value** box, or click  and select a variable you want to assign as the value.

For more information about adding metadata objects to variables, see “[Tagging Variables with Metadata](#)” on page [74](#).

9. Click **Next**.

The **Variables from XML Tags** dialog box opens and displays the hierarchy of tags from the XML data or schema. The hierarchy includes attributes from tags, listed as children of the tags in which they appear and preceded with "(A)."

10. Use the options on the dialog box to specify which variables Design Manager will create and to edit the names and data types of the created variables:

| To | Do this |
|--|--|
| Add the same prefix to the name of each variable created | In the Variable name prefix box, enter the prefix to add, and then click Apply . |

| To | Do this |
|---|---|
| Ignore a tag or a branch when creating variables | <ul style="list-style-type: none"> a. In the XML tag tree, highlight the tag you want to ignore or the parent tag of a branch you want to ignore. b. Select the Ignore check box. |
| Create variables from only one branch of the XML data or schema | <ul style="list-style-type: none"> a. In the XML tag tree, highlight the parent tag of the branch from which you want to create variables. b. Click Ignore all but this branch. |
| Reset the list of ignored tags | <p>Click Clear all ignore flags.</p> <p>This clears the Ignore check box for all tags in the list.</p> |
| Edit the name of a variable created from a tag | <ul style="list-style-type: none"> a. In the XML tag tree, highlight the tag for which you want to change the variable name. b. In the Name box, enter a name for the new variable. |
| Edit the data type of a variable created from a tag | <ul style="list-style-type: none"> a. In the XML tag tree, highlight the tag for which you want to change the data type. b. From the Type list, select a data type for the new variable. |
| Create an array variable from a tag | <ul style="list-style-type: none"> a. In the XML tag tree, highlight the tag for which you want to create an array variable. b. Select the Array check box. |
| Automatically select the parent tag of a selected tag | Beside the Hierarchy of selected tag box, click Up . |

11. In the **Duplicate names** area, review the list of potential variable names that are currently duplicated. If you do not resolve duplicate names, only one variable will be created for each set of duplicates. To resolve duplicate names and make sure that a variable is created for each desired tag, you must do one of the following:

| To | Do this |
|--|---|
| Resolve duplicate names manually | Select each tag that results in a duplicate name and enter a new name in the Name box. |
| Resolve duplicate names automatically using the names of parent tags | <p>Click Resolve using parent names.</p> <p>This adds the name of the parent tag and an underscore character to the beginning of each duplicated name.</p> |

12. Click **Finish**.

The **Variables from XML Tags** dialog closes and Design Manager creates the specified variables.

3.2.6 Populating a String Variable Using a String Lookup Table

Some data files include codes or abbreviations to represent long data strings. Depending on the number of abbreviations, how often they change, and when they are needed, you have several options for substituting the complete strings for the abbreviations. You can use an initialization file to load all of the complete strings at the beginning of processing and store them in memory, you can use a reference file to look up the complete strings during processing, or, if you have just a few abbreviations, you can create a string lookup table.

A string lookup table contains all of the abbreviations and complete strings available in the data file. During processing, the value of the variable automatically updates to the complete string when the engine reads one of the abbreviations. For example, if you provide multiple levels of service, you can create a variable called 'ServiceLevel' with a string lookup table similar to the following:

```
3 = Full Service Level 2 = Average Service Level 1 = Minimal Service Level
```

If the value of the variable is set to ASL, then the value automatically expands to Average Service Level.

To populate a string variable using a string lookup table:

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Basic** tab.
3. From the **Data type** list, select **String**.
4. From the **Source** list, select **File only**.
5. On the **String Lookup** tab, select the **Automatically convert to string values from** check box.
6. To specify the location of the string lookup information, you must do one of the following:

| To | Do this |
|---|--|
| Define a static list within the variable properties | <ol style="list-style-type: none">a. From the Automatically convert to string values from list, select Static list.b. Click +. The Lookup String dialog box opens.c. In the Identifier box, enter the identifier the engine uses to look up the string.d. In the Value box, enter the string value that will replace the identifier in the final output.e. Click OK. The Lookup String dialog box closes and the string lookup appears in the box. |

| To | Do this |
|--|--|
| Search for the information in array variables using a linear search | <ol style="list-style-type: none">From the Automatically convert to string values from list, select Array variables.From the Array containing match values box, select the array variable that contains the identifiers used for the string lookup.From the Array containing new values box, select the array variable that contains the string values that will replace the identifiers. |
| Search for the information in sorted array variables using a binary search | <ol style="list-style-type: none">From the Automatically convert to string values from list, select Sorted array variables.From the Array containing match values box, select the array variable that contains the identifiers used for the string lookup.From the Array containing new values box, select the array variable that contains the string values that will replace the identifiers. |

7. To specify the action that occurs if an identifier is not defined, select one of the following options from the **Action if not found** list:
 - **Clear**—The engine discards the value and continues processing.
 - **Keep**—The engine retains the value found in the data file and continues processing.
 - **Clear, issue message**—The engine discards the value, you receive a message, and the engine continues processing.
 - **Keep, issue message**—The engine retains the value found in the data file, you receive a message, and the engine continues processing.
 - **Clear, generate error**—The engine discards the value and stops processing.
 - **Keep, generate error**—The engine retains the value found in the data file and stops processing.
8. From the **Edit** menu, select **Save**.

3.2.7 Adding Variables to an Application

You must add variables to your application to make sure the application includes the correct version of the variable. For example, you might have variables that share the same name but reside in different folders or you have variables in formulas that don't appear elsewhere in the application. When you package for any target, such as campaigns or documents, the package file always includes the variables you have included in the application.

To add a variable to an application:

1. In Design Manager, from the Library, drag an application to the Property Panel.
2. Go to the **Variables** tab.

3. Under the **Application variables** box, click .

The **Select Variable** dialog box opens.

4. Click the variable you want to add.
5. Click **OK**.

The **Select Variable** dialog box closes and the variable is added to the **Application variables** box.

6. From the **Edit** menu, select **Save**.

3.2.8 Specifying How Array Variable Elements are Displayed

If your design includes an array variable, you can specify how the elements in the array are displayed. This can be useful for situations like defining how array elements should be arranged during data aggregation, or controlling how the engine selects the correct external file when using a placeholder variable.

To specify how array elements are displayed:

1. Open a paragraph in Designer.
2. Insert the variable into the text.
3. Right-click the variable and select **Variable Properties**.
4. Click the **Variable Use** tab.

5. In the **Array element to use** list, select one of the following options:

| To | Do this |
|--|--|
| Automatically display an array element for each occurrence of the variable | Select Automatic . |
| Display a specific element in the array | <ol style="list-style-type: none">Select Specified.In the adjacent box, select the specific element number you want to use for this variable. <p>This option should only be used with a static array.</p> <p>Note: If you are using a placeholder static array variable to import external files, you should use this option. If you do not use this option, the engine inserts the first mapped external file listed in the data file.</p> |
| Display all instances of the array in the data file for each occurrence of the variable, including blank spaces | Select All, blanks . |
| Display all instances of the array in the data file for each occurrence of the variable, and separate each element in content using a line break (soft return). | Select All, line breaks . |
| Display all instances of the array in the data file for each occurrence of the variable, and separate each element in the content using a paragraph break (hard return). | Select All, paragraphs . |
| Display all instances of the array in the data file for each occurrence of the variable, and list them in order. | Select All, print as list . |

6. Click **OK**.

3.3 Formatting Variable Values in Output

Variable values, and the output format of those values, come directly from the data in a data source. However, in some cases, the format of the data in the data source is not the format you want in the final output. You are not required to update the data source to create the format you want; instead, you can customize the output format directly on the variable. This ability is especially useful if you have mapped the same variable to multiple data files. No matter which data file an application uses, the output will still have the same format.

You select the format of the variable values using the data type associated with the variable. To format variable values in the output, complete the following tasks as needed:

- “Formatting Boolean Values” below
- “Formatting Currency Values” on the next page
- “Formatting Date Values” on page 86
- “Formatting Floating Values” on page 87
- “Formatting Integer Values” on page 91
- “Formatting String Values” on page 94
- “Formatting Tagged Text Values” on page 96

3.3.1 Formatting Boolean Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select one of the following options:
 - **T/F**—Return T when the condition is true or F when the condition is false.
 - **Y/N**—Return Y when the condition is true or N when the condition is false.
 - **1/0**—Return 1 when the condition is true or 0 when the condition is false.
 - **True/False**—Return True when the condition is true or False when the condition is false.
 - **Yes/No**—Return Yes when the condition is true or No when the condition is false.
 - **General Number**—Return a number based on the condition.
4. To prevent the variable value from being split onto multiple lines when it is substituted into a paragraph, select the **Do not break within variable** check box.
5. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see the "Configuring a Language Object to Use Complex Text Layout" section in *System Administration* in the Exstream Design and Production documentation.

6. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following:
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens.
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
7. From the **Edit** menu, select **Save**.

3.3.2 Formatting Currency Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select one of the following options:
 - **Custom**—Create and use a customized formatting.
 - **Use Locale Specification**—Format the data based on the locale of the customer.
 - **General Number**—Use simple numbers.
 - **Fixed Decimal**—Add zeros to pad the number of digits after the decimal point.
 - **Fixed Decimal with Currency**—Insert a fixed decimal point and the proper currency symbol for the locale.
 - **Fixed or Integer with Currency**—Insert a whole number if the number does not have a fraction and inserts the proper currency symbol for the locale. If the number has a fraction, the engine formats it as a floating number and inserts the proper currency symbol for the locale.
 - **Significant Decimal**—Insert the least number of decimal values possible after the whole number.
 - **Fixed or Integer**—Insert a whole number if the number does not have a fraction. If the number has a fraction, the engine formats it as a floating number.

4. To further customize the variable, complete one of the following tasks based on what you select from the **Output format** list:

| To | Do this |
|------------------------|---|
| Create a custom format | <ol style="list-style-type: none">a. From the Output format list, select Custom.b. In the Custom format string box, enter a custom format string, or complete the following steps:<ol style="list-style-type: none">i. Right-click the Custom format string box and select Format. The Select Formatting String dialog box opens.ii. From the Data type list, select the data type that you want to use to filter the list of available custom formatting options.iii. In the box, select the custom format that you want to insert.iv. Click OK. The Select Formatting String dialog box closes and the format appears in the Custom format string box.v. Repeat step i through step iv until the custom format is defined. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: Some custom formats require you to enter text. For example, if you selected "ABC" replace ABC with your custom text.</p></div> <p>For more information about creating a custom format, see "Creating a Custom Format String" on page 97.</p> |
| Format the number | <ol style="list-style-type: none">a. Select the Thousands check box and select the thousands separator or enter a custom separator.b. From the Negative format list, select the negative number formatting.c. In the Number of digits box, enter the number of digits to appear after the decimal point. If you selected Significant decimal from the Output format list, the number you enter represents the maximum number of decimal places to include in the output. Additional digits exceeding the number are rounded, and the engine removes any zeros at the end of the decimal value. For example, if you enter 2 in the Number of digits box, 5.303 appears as 5.3 and 5.003 appears as 5.d. From the Decimal symbol list, select the decimal symbol or enter a custom decimal symbol. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: These options also affect currency formats and work with the location of the currency symbol.</p></div> |

| To | Do this |
|----------------------------|---|
| Format the currency symbol | <p>a. From the Output format list, select one of the currency formats. The Currency Format area becomes available.</p> <p>b. Select the Use special formatting check box. The remaining options in the Currency Format area become available.</p> <p>c. Specify the currency symbol by doing one of the following:</p> <ul style="list-style-type: none">• In the Currency Symbol box, enter the currency symbol to use.• Select the Use locale currency symbol check box to use the currency symbol specified in the locale properties. <p>d. Select one or more of the following check boxes to customize the currency symbol location. The options available vary based on the option selected from the Negative format list.</p> <ul style="list-style-type: none">• Place the currency symbol after the number• Place the currency symbol after the negative symbol• Place the currency symbol inside the parenthesis <p>The formatting in the Preview area updates.</p> <p>For more information about configuring a locale, see <i>System Administration</i> in the Exstream Design and Production documentation.</p> |

5. To prevent the variable value from being split onto multiple lines when it is substituted into a paragraph, select the **Do not break within variable** check box.
6. To override values of zero with a custom format, you must do the following:
 - a. Select the **Replace zero** check box.
 - b. In the adjacent box, enter the text you want to use to override values of zero.
 - c. To change the reading order or insert Unicode characters, right-click the box and select one of the following options:
 - **Right to left Reading order**
 - **Show Unicode control characters**
 - **Insert Unicode control character**
7. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format

for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see the "Configuring a Language Object to Use Complex Text Layout" section in *System Administration* in the Exstream Design and Production documentation.

8. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
9. From the **Edit** menu, select **Save**.

3.3.3 Formatting Date Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select the format option.
4. If you selected **Custom** from the **Output format** list, you must enter a custom format string in the **Custom format string** box, or do the following:
 - a. Right-click the **Custom format string** box and select **Format**.
The **Select Formatting String** dialog box opens.
 - b. From the **Data type** list, select the data type that you want to use to filter the list of available custom formatting options.
 - c. In the box, select the custom format that you want to insert.
 - d. Click **OK**.
- The **Select Formatting String** dialog box closes and the format appears in the **Custom format string** box.
- e. Repeat step a through step d until the custom format is defined.

Note: Some custom formats require you to enter text. For example, if you select "ABC" replace ABC with your custom text.

For more information about creating a custom format, see [“Creating a Custom Format String” on page 97](#).

5. To prevent the variable value from being split onto multiple lines when it is substituted into a paragraph, select the **Do not break within variable** check box.
6. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see the “Configuring a Language Object to Use Complex Text Layout” section in *System Administration* in the Exstream Design and Production documentation.

7. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following:
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens.
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
8. From the **Edit** menu, select **Save**.

3.3.4 Formatting Floating Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select one of the following options:

- **Custom**—Create and use a customized formatting.
- **Use Locale Specification**—Format the data based on the locale of the customer.
- **General Number**—Use simple numbers.
- **Percentage**—Add a percent (%) symbol to the end of the number.
- **Percentage x 100**—Add a percent (%) symbol at the end of the number and multiply by 100. This option lets you format fractions as percents. For example, .75 appears as 75%
- **Fixed Decimal**—Add zeros to pad the number of digits after the decimal point.
- **Fixed Decimal with Currency**—Insert a fixed decimal point and the proper currency symbol for the locale.
- **Fixed or Integer with Currency**—Insert a whole number if the number does not have a fraction and insert the proper currency symbol for the locale. If the number has a fraction, the engine formats it as a floating number and inserts the proper currency symbol for the locale.
- **Significant Decimal**—Insert the least number of decimal values possible after the whole number.
- **Fixed or Integer**—Insert a whole number if the number does not have a fraction. If the number has a fraction, the engine formats it as a floating number.
- **Absolute Value**—Convert the values to the absolute value. This option lets you produce a negative number without a negative sign. For example, both 1 and -1 appear as 1.

4. To further customize the variable, complete one of the following tasks based on what you select from the **Output format** list:

| To | Do this |
|------------------------|--|
| Create a custom format | <ol style="list-style-type: none">a. From the Output format list, select Custom.b. In the Custom format string box, enter a custom format string, or complete the following steps:<ol style="list-style-type: none">i. Right-click the Custom format string box and select Format. The Select Formatting String dialog box opens.ii. From the Data type list, select the data type that you want to use to filter the list of available custom formatting options.iii. In the box, select the custom format that you want to insert.iv. Click OK. The Select Formatting String dialog box closes and the format appears in the Custom format string box.v. Repeat step i through step iv until the custom format is defined. <p>Note: Some custom formats require you to enter text. For example, if you selected "ABC" replace ABC with your custom text.</p> <p>For more information about creating a custom format, see "Creating a Custom Format String" on page 97.</p> |
| Format the number | <ol style="list-style-type: none">a. Select the Thousands check box and select the thousands separator or enter a custom separator.b. From the Negative format list, select the negative number formatting.c. In the Number of digits box, enter the number of digits to appear after the decimal point. If you selected Significant decimal from the Output format list, the number you enter represents the maximum number of decimal places to include in the output.d. From the Decimal symbol list, select the decimal symbol or enter a custom decimal symbol. <p>Note: These options also affect currency formats and work with the location of the currency symbol.</p> |

| To | Do this |
|--|---|
| Format the currency symbol in a monetary value | <p>a. From the Output format list, select one of the currency formats. The Currency Format area becomes available.</p> <p>b. Select the Use special formatting check box. The remaining options in the Currency Format area become available.</p> <p>c. Specify the currency symbol by doing one of the following:</p> <ul style="list-style-type: none">• In the Currency Symbol box, enter the currency symbol to use.• Select the Use locale currency symbol check box to use the currency symbol specified in the locale properties. <p>d. Select one or more of the following check boxes to customize the currency symbol location. The options available vary based on the option selected from the Negative format list.</p> <ul style="list-style-type: none">• Place the currency symbol after the number• Place the currency symbol after the negative symbol• Place the currency symbol inside the parenthesis <p>The formatting in the Preview area updates.</p> <p>For more information about configuring a locale, see <i>System Administration</i> in the Exstream Design and Production documentation.</p> |

5. To override values of zero with a custom format, you must do the following:
 - a. Select the **Replace zero** check box.
 - b. In the adjacent box, enter the text you want to use to override values of zero.
 - c. To change the reading order or insert Unicode characters, right-click the box and select one of the following options:
 - **Right to left Reading order**
 - **Show Unicode control characters**
 - **Insert Unicode control character**
6. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see the "Configuring a Language Object to Use Complex Text Layout" section in *System Administration* in the Exstream Design and Production documentation.

7. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following:
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
8. From the **Edit** menu, select **Save**.

3.3.5 Formatting Integer Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select one of the following options:
 - **Custom**—Create and use a customized formatting.
 - **General Number**—Use simple numbers.
 - **Percentage**—Add a percent (%) symbol to the end of the number.
 - **Text Upper**—Convert integer values to uppercase words. For example, 1 appears as ONE.
 - **Text Mixed**—Convert integer values to mixedcase words. For example, 1 appears as One.
 - **Text Lower**—Convert integer values to lowercase words. For example, 1 appears as one.
 - **Alpha Upper**—Convert integer values to uppercase alpha characters. Each integer is associated with a letter of the alphabet, all uppercase. If you use a number greater than 26, the alphabet starts over. For example, 1 appears as A and 27 appears as AA.
 - **Alpha Lower**—Convert integer values to lowercase alpha characters. Each integer is associated with a letter of the alphabet, all lowercase. If you use a number greater than 26, the alphabet starts over. For example, 1 appears as a and 27 appears as aa.
 - **Roman Upper**—Convert integer values to uppercase Roman numerals.

- **Roman Lower**—Convert integer values to lowercase Roman numerals.
 - **Fixed Decimal with Currency**—Insert a fixed decimal point and the proper currency symbol for the locale.
 - **Fixed or Integer with Currency**—Insert a whole number if the number does not have a fraction and insert the proper currency symbol for the locale. If the number has a fraction, the engine formats it as a floating number and inserts the proper currency symbol for the locale.
 - **Absolute Value**—Convert the values to the absolute value. This option lets you produce a negative number without a negative sign. For example, both 1 and -1 appear as 1.
4. To further customize the variable, complete one of the following tasks based on what you select from the **Output format** list:

| To | Do this |
|------------------------|--|
| Create a custom format | <ol style="list-style-type: none">a. From the Output format list, select Custom.b. In the Custom format string box, enter a custom format string, or complete the following steps:<ol style="list-style-type: none">i. Right-click the Custom format string box and select Format. The Select Formatting String dialog box opens.ii. From the Data type list, select the data type that you want to use to filter the list of available custom formatting options.iii. In the box, select the custom format that you want to insert.iv. Click OK. The Select Formatting String dialog box closes and the format appears in the Custom format string box.v. Repeat step i through step iv until the custom format is defined. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: Some custom formats require you to enter text. For example, if you selected "ABC" replace ABC with your custom text.</p></div> <p style="margin-top: 20px;">For more information about creating a custom format string, see "Creating a Custom Format String" on page 97.</p> |

| To | Do this |
|--|--|
| Format the currency symbol in a monetary value | <ol style="list-style-type: none">a. From the Output format list, select one of the currency formats. The Currency Format area becomes available.b. Select the Use special formatting check box. The remaining options in the Currency Format area become available.c. Specify the currency symbol by doing one of the following:<ul style="list-style-type: none">• In the Currency Symbol box, enter the currency symbol to use.• Select the Use locale currency symbol check box to use the currency symbol specified in the locale properties.d. Select one or more of the following check boxes to customize the currency symbol location. The options available vary based on the option selected from the Negative format list.<ul style="list-style-type: none">• Place the currency symbol after the number• Place the currency symbol after the negative symbol• Place the currency symbol inside the parenthesis The formatting in the Preview area updates. <p>For more information about configuring a locale, see <i>System Administration</i> in the Exstream Design and Production documentation.</p> |

5. To prevent the variable value from being split onto multiple lines when it is substituted into a paragraph, select the **Do not break within variable** check box.
6. To override values of zero with a custom format, you must do the following:
 - a. Select the **Replace zero** check box.
 - b. In the adjacent box, enter the text you want to use to override values of zero.
 - c. To change the reading order or insert Unicode characters, right-click the box and select one of the following options:
 - **Right to left Reading order**
 - **Show Unicode control characters**
 - **Insert Unicode control character**
7. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format

for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see the "Configuring a Language Object to Use Complex Text Layout" section in *System Administration* in the Exstream Design and Production documentation.

8. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
9. From the **Edit** menu, select **Save**.

3.3.6 Formatting String Values

1. From the Variable Palette, drag the variable to the Property Panel.
2. Go to the **Output Format** tab.
3. From the **Output format** list, select one of the following options:
 - **Custom**—Create and use a customized formatting.
 - **Keep Blanks**—Preserve spaces as they appear in the string.
 - **Trim Blanks**—Remove all of the spaces found before the first character and after the last character of the string.
 - **Lower, Keep Blanks**—Convert the string to all lowercase characters and preserve all of the spaces as they appear in the string.
 - **Lower + Exceptions, Keep Blanks**—Convert the string to all lowercase characters, except those words found in an exceptions file, and preserve all of the spaces as they appear in the string.
 - **Upper, Keep Blanks**—Convert the string to all uppercase characters and preserve all of the spaces as they appear in the string.
 - **Upper + Exceptions, Keep Blanks**—Convert the string to all uppercase characters, except those words found in an exceptions file, and preserve all of the spaces as they appear in the string.

- **InitCap, Keep Blanks**—Convert the string so each word begins with an uppercase character and preserve all of the spaces as they appear in the string.
 - **InitCap, Trim Blanks**—Convert the string so each word begins with an uppercase character and remove all of the spaces found before the first character and after the last character of the string.
 - **InitCap + Exceptions, Keep Blanks**—Convert the string so each word begins with an uppercase character, except those words found in an exceptions file, and preserve all of the spaces as they appear in the string.
 - **Sentence, Keep Blanks**—Convert the string so only the first word of a sentence begins with an uppercase character and preserve all of the spaces as they appear in the string.
 - **Sentence + Exceptions, Keep Blanks**—Convert the string so only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and preserve all of the spaces as they appear in the string.
 - **Exceptions Only, Keep Blanks**—Convert the words found in the exception file to match their format in the file and preserve all of the spaces as they appear in the string.
 - **Phone Number (999) 999-9999**—Convert string values to a telephone number with the area code in parentheses and a hyphen between the first three and last four digits of the number.
 - **Phone Number (999) 999 9999**—Convert string values to a telephone number with the area code in parentheses and a space between the first three and last four digits of the number.
 - **Phone Number 999 999-9999**—Convert string values to a telephone number with the area code without parentheses and a hyphen between the first three and last four digits of the number.
 - **Phone Number 999 999 9999**—Convert string values to a telephone number with the area code without parentheses and a space between the first three and last four digits of the number.
4. If you selected **Custom** from the **Output format** list, you must enter a custom format string in the **Custom format string** box, or do the following:
- a. Right-click the **Custom format string** box and select **Format**.

The **Select Formatting String** dialog box opens.
 - b. From the **Data type** list, select the data type that you want to use to filter the list of available custom formatting options.
 - c. In the box, select the custom format that you want to insert.
 - d. Click **OK**.

The **Select Formatting String** dialog box closes and the format appears in the **Custom format string** box.

- e. Repeat step a through step d until the custom format is defined.

Note: Some custom formats require you to enter text. For example, if you selected "ABC" replace ABC with your custom text.

For more information about creating a custom format, see "[Creating a Custom Format String](#)" on the next page.

5. To prevent the variable value from being split onto multiple lines when it is substituted into a paragraph, select the **Do not break within variable** check box.
6. If this variable is being used on a language layer that uses the Arabic script (and you have complex text layout enabled), but you do not want numbers in variable values to be converted to that script, then you should select the **Do not convert numbers on language layers** check box. (This setting is useful if a variable value contains numbers that must not be converted, such as those in email addresses.)

Selecting this check box tells the engine to ignore the **Numerical Digits** setting on the language object associated with each language layer, and instead use the original format for numbers in the variable value. Note that the **Do not convert numbers on language layers** setting does not apply to the default language layer.

For more information about the **Numerical Digits** setting, see *System Administration* in the Exstream Design and Production documentation.

7. If you are working in a DBCS application and you want to control the width of characters in the output, you must do the following
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
The **Conversion Properties** dialog box closes.
 - c. Click **OK**.
8. From the **Edit** menu, select **Save**.

3.3.7 Formatting Tagged Text Values

If you are using tagged text variables, the formatting is determined by the tag set in the application. To format a tagged text variable, you must add a tag set to an application.

For more information about tag sets, see *System Administration* in the Exstream Design and Production documentation.

To format a tagged text variable:

1. In Design Manager, from the Library, drag an application to the Property Panel.
2. Click the **Documents** tab.
3. In the **Tag set** box, select the tag set you want to add to the application. The options vary based on the tag sets you have available.
4. From the **File** menu, select **Save**.

3.3.8 Creating a Custom Format String

If you selected **Custom** from the **Output format** list for a currency, date, integer, or string value, you can use the following strings in the **Custom format string** box to manually construct a custom format string:

- 0—Display a digit or else pad with a zero.
- #—Display a digit or else nothing.
- .—Display a decimal placeholder.
- %—Multiply the value by 100 and display the number as a percentage.
- ,—Display a thousands separator.
- :—Display a time separator.
- /—Display a date separator.
- E-, E+, e-, or e+—Display the number in scientific format.
- --Display a minus sign.
- +—Display a plus sign.
- \$—Display a dollar sign.
- (or)—Display a left or right parenthesis.
- \—Display the next literal character.
- "ABC"—Display the characters ABC or any other characters within the quotation marks.
- @—Display a character or else a space.
- &—Display a character or else nothing.
- <—Force all characters to appear in lowercase.
- >—Force all characters to appear in uppercase.

- !—Force left to right fill of placeholders.
- a/p—Display the hour using the 12-hour clock and a or p.
- am/pm—Display the hour using the 12-hour clock and am or pm.
- A/P—Display the hour using the 12-hour clock and A or P.
- AM/PM—Display the hour using the 12-hour clock and AM or PM.
- AMPM—Display the AM or PM for the specified hour.
- c—Display the date and time as m/d/yy h:nn:ss.
- d—Display the day as a number with no leading zero.
- dd—Display the day as a number with a leading zero.
- ddd—Display the abbreviated name of the day of the week.
- DDD—Display the abbreviated name of the day of the week in uppercase.
- dddd—Display the full name of the day of the week.
- DDDD—Display the full name of the day of the week in uppercase.
- ddddd—Display the date as m/d/yy.
- dddddd—Display the date as mmmm d, yyyy.
- h—Display the hour as a number without a leading zero.
- hh—Display the hour as a number with a leading zero.
- m—Display the month as a number without a leading zero.
- mm—Display the month as a number with a leading zero.
- mmm—Display the abbreviated name of the month.
- MMM—Display the abbreviated name of the month in uppercase.
- mmmm—Display the full name of the month.
- MMMM—Display the full name of the month in uppercase.
- n—Display the minute as a number without a leading zero.
- nn—Display the minute as a number with a leading zero.
- q—Display the calendar quarter as a number.
- s—Display the second as a number without a leading zero.
- ss—Display the second as a number with a leading zero.
- tttt—Display the complete time as h:nn:ss.
- y—Display the day of the year as a number.

- yy—Display the last two digits of the year.
- yyyy—Display the year.

You can also use these strings to construct a format expression for the Format built-in function.

For more information about using the Format built-in function, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

3.4 Testing Variables

An essential part of any design process is testing the design. During the testing phase, you can create reports and run tests to ensure that the engine creates the output you expect. You can test an application before you put it into production or throughout the entire application design process. Since variables control the dynamic portions of a design, it is important to make sure that they are used correctly. Exstream Design and Production offers tools that let you test the variables you have included in an application.

To test variables, complete the following tasks as needed:

- “[Testing a Variable to Ensure Correct Data Population](#)” below
- “[Creating a Report to List the Variables Used in an Application](#)” on page 101

3.4.1 Testing a Variable to Ensure Correct Data Population

Most commonly, variables populate an application with data from a data file. You can use a variable in a rule to include or exclude an object, or directly on a page to personalize a design. Since variables can have multiple uses in the application design, it is important to make sure that the variable populates the application with the correct data. If the variable is populated with incorrect data, the personalization might not work or the wrong objects might be included. To ensure that the variables create the output you expect, you can test the variable to ensure that it populates the application with the correct data.

To test the population of a variable, you must select an application that includes a data file mapped with the variable you want to test. When you run the test, the data in the data file populates the variable, which lets you verify information such as the possible values of the variable.

Note that this testing is for design purposes only. Variable testing in Design Manager is meant to test variables that are mapped to a data file. This test does not verify the population of variables that can change their value during an engine run. For example, suppose that you want to verify that a table is being populated with currency values. Because the engine does not complete the composition steps necessary to create pages or process output queues, values that are

calculated during processing might not be accurate. Results for page-counting variables and other processing-dependent variables are not included. In addition, applications might have code or formulas that the engine calculates only at run time. These production-run calculations can have effects on the output that you might not see while testing variables.

If you are testing the variables for a schema model data file, reference file sections appear in the order they are read, based on the node structure. Sections and nodes are indicated in the test results.

To test a variable to ensure correct data population:

1. From the Variable Palette, drag the variable to the Property Panel.
2. On the Standard toolbar, click  The **Data Dictionary Test Results** dialog box opens.
3. In the **What to test** area, select an application from the **Application** box.
4. Select the **Limit sets** check box and complete the three boxes to limit the number of customers used to perform the test.
5. To select which variables to test, you must do one of the following:

| To | Do this |
|---|---|
| Test a specific variable | <ol style="list-style-type: none">a. Select the Selected only check box.b. Select the variable to test. |
| Test all of the variables in the selected application | Clear the Selected only check box. |

6. In the **What results to show** area, select one or more of the following check boxes to specify the type of information to return:
 - **User values**—Include the values of variables that have the **Constant user value** option selected from the **Source** list on the **Basic** tab of the variable properties. This option is active only if the **Selected Only** check box is cleared.
 - **System variables**—Include the values of system variables. This option is active only if the **Selected Only** check box is cleared.
 - **Initial**—Include the values of variables encountered before the first customer is processed.
 - **Customers**—Include values of variables during customer processing.
 - **Final**—Include the values of variables after the last customer has been processed.
 - **Sections and Nodes**—Include values of variables at each data section and node.
7. Click **Go**.

The engine runs and the test results appear in the lower portion of the **Data Dictionary Test Results** dialog box.

8. To review the properties of a variable, do the following:
 - a. In the **Variable** area, the **Results** area, or the **Messages** area, select a result.
 - b. Click **Find**.

The associated variable opens in the Property Panel.

3.4.2 Creating a Report to List the Variables Used in an Application

Before putting an application into production, you can run a report that lists all of the variables used in the design of an application. Variables that are mapped to a data file in the application, but are not used in the application design, are not included in the report. Depending on when you create the report, you can use the report to select which data areas to map in input data files (for example, customer driver files or reference files). For example, if you create the variable report before you map a customer driver file, you can use the report to map only the data areas required to run the application, which could save you processing time and memory when running the engine. If you have already mapped the data file, you can use this report to compare the variables used in the application with the variables mapped to the data file.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

If you have licensed the XML/JSON Input module, you can use a variable report to create a sample XML data layout. The sample data layout includes the variables used in the design which have values that must come from a data file.

For more information about creating a sample XML data layout, see “[Using a Variable Report to Create a Sample XML Data Layout](#)” on page 210.

To create a report to view the variables used in an application:

1. In Design Manager, in the Library, right-click an application and select **Variable Report**.

The variable report opens in the Edit Panel.

2. To filter the report by approval status, design user, or effective dates, do the following:

- a. Click **Object Selection Settings**.

The **Object Selection Version Settings** dialog box opens.

- b. Use the options on the dialog box to specify which variables are to be included in the report:

- i. From the **Version method** list, select **Version status**.

The **Includes selected status and any statuses above it** slider area appears.

- ii. Move the slider to one of the following options:

- **Approved**—Includes all objects with a status of Approved or Archived. If you select **Approved**, you can choose to substitute quick fix versions by selecting the **Substitute Quick Fix versions** check box.
- **Submitted for approval**—Includes all objects with a status of Submitted for Approval, Approved, and Archived
- **Work in progress**—Includes all objects with a status of Work in Progress, Submitted for Approval, Approved, and Archived
- **Rejected**—Includes all objects with a status of Rejected, Work in Progress, Submitted for Approval, Approved, and Archived

- iii. If you want the report to include only objects from a specific user, and you selected **Submitted for Approval**, **Work in Progress**, or **Rejected** on the **Includes selected status and any statuses above it** slider, select the **Include versions from the following user** check box.

- iv. Under the **Include versions from the following user** check box, click .

The **Select Design User** dialog box opens.

- v. From the list of design users, select a design user.

- vi. Click **OK**.

The design user that you selected appears in the box under the **Include versions from the following user** check box.

- vii. From the **Effective date** list, select how effectiveness dates affect which objects can be selected for packaging:

- **As of Now**—Selects the latest valid version of the objects as of the current date (and time, if used)
- **As of Date**—Selects the latest valid versions of the objects as of a specified date (and time, if used). Select the date and time from the adjacent **To** box.
- **Date Range**—Selects the valid version of objects as of a range of dates (and times, if used). Select the date and time from the adjacent **From** and **To** boxes.

- viii. Click **OK**.

The **Object Selection Version Settings** dialog box closes and the settings you selected appear in the **Version** and **Effective** boxes.

3. To save the report, do the following:

- a. Right-click the Edit Panel and select **Export List**.

The **Save As** dialog box opens.

- b. In the **Save in** box, select the folder in which to save the variable report.

- c. In the **File name** box, enter the name of the variable report.

- d. From the **Save as type** list, select one of the following options:

- **Comma separated values (*.csv)**—Save the report as a comma delimited text file.

- **Tab delimited text (*.txt)**—Save the report as a tab delimited text file.

- e. Click **Save**.

4. To print the report, do the following:

- a. Right-click the report and select **Print List**.

The **Print** dialog box opens.

- b. Select the printer and the printer options you want.

- c. Click **OK**.

3.5 System Variables

System variables let you use system information, such as the information stored in the design environment or the information generated as part of the packaging process, to complete tasks such as monitoring and controlling an engine run, formatting designs, or organizing a customer mailing. System variables are always stored under the **Data Dictionary** heading under the Exstream Design and Production root folder; however, you can divide system variables into categories.

This chapter discusses the following topics:

- “[Application Design Related System Variables](#)” on the next page
- “[Counting and Numbering Related System Variables](#)” on page 106
- “[Customer Related System Variables](#)” on page 113

- “[Data Related System Variables](#)” on page 115
- “[Date Related System Variables](#)” on page 117
- “[Design Related System Variables](#)” on page 118
- “[High-Volume Output Processing Related System Variables](#)” on page 119
- “[Marketing Related System Variables](#)” on page 124
- “[Production Related System Variables](#)” on page 126
- “[Real-time and SOAP-Related System Variables](#)” on page 127

In this chapter, if a system variable name is in bold, you can calculate its value using a formula or rule.

For more information about creating a formula or a rule, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

3.5.1 Application Design Related System Variables

The system variables in the following table are related to the objects required to create an application: applications, documents, and pages. They provide information such as the number of documents or pages in an application, whether the content on a page flows to another page, or the names of all of the documents in an application.

For more information about application design, see *Designing Customer Communications* in the Exstream Design and Production documentation.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Application design related system variables

| Variable name | Description |
|-----------------------|--|
| SYS_CurrentProfile | The name of the recipient profile used for the current recipient copy |
| SYS_CustomerDocuments | The total number of documents included in the customer |
| SYS_DocFormattedValue | <p>The formatted string used as the document or chapter number in a table of contents entry or on a page</p> <p>The value increments for each document in a customer and restarts at the beginning of a new customer or when a document specifies a restart.</p> |
| SYS_DocPrintedValue | <p>The value to use as the document or chapter number on the current page</p> <p>The number increments for each document in a customer and restarts at the beginning of a new customer or when a document specifies a restart.</p> |

Application design related system variables, continued

| Variable name | Description |
|---------------------------------|--|
| SYS_DocumentNames | The names of the documents included in the customer |
| SYS_PageFlows | <p>The value used to identify whether the current page has content that flows to another page</p> <p>If the 'SYS_PageFlows' system variable has a value other than 0, the page flows. A 0 indicates that the page does not flow.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Tip: You can use 'SYS_PageFlows' system variable with a rule to specify when continued should be placed at the bottom of a page. </div> |
| SYS_PageFormattedValue | <p>The formatted string to use as the page number in a table of contents entry or on a page</p> <p>The page number might differ from page count because the page count might have been restarted or because of blank pages. Banner pages are not counted.</p> |
| SYS_PageInDocument | <p>The sequence number of the current counted page within the document. You can use the application properties to exclude some types of pages from the page count. Banner pages are not counted.</p> <p>The value of the 'SYS_PageInDocument' system variable might vary from the value of the 'SYS_SheetInDocument' system variable if duplex pages are included in a document.</p> |
| SYS_PagePhysicalInDocument | <p>The sequence number of the current physical page within the document</p> <p>The number of physical pages might differ from counted pages because of the page counting methods set on the application.</p> |
| SYS_PagePrintedValue | <p>The value to use as the page number of the current page</p> <p>The page number might differ from page count because the page count might have been restarted or because of blank pages. Banner pages do not have a printed value.</p> |
| SYS_PageStart | The initial number of pages included in the document before pages are excluded for reasons such as rules or marketing page limits |
| SYS_PageTotalInDocument | <p>The total number of counted pages included in the document</p> <p>The value of the 'SYS_PageTotalInDocument' system variable might vary from the value of the 'SYS_SheetTotalInDocument' system variable when duplex pages are included in a document.</p> |
| SYS_PageTotalPhysicalInDocument | <p>The total number of physical pages included in the document</p> <p>The number of physical pages might differ from counted pages because of the page counting methods set on the application.</p> |
| SYS_PageTotalPrinted | <p>The page number of the last page in the document</p> <p>The value of the 'SYS_PageTotalPrintedInDocument' system variable might vary from the value of the 'SYS_PageTotalPhysicalInDocument' system variable when page counts reset at the start of a document.</p> |
| SYS_PaperTypes | The names of the paper types used in an application during an engine run |

Application design related system variables, continued

| Variable name | Description |
|----------------------------|---|
| SYS_RecipientProfiles | The names of the recipient profiles used in an application during an engine run |
| SYS_SheetInDocument | The sequence number of the current sheet within the document The value of the 'SYS_SheetInDocument' system variable might vary from the value of the 'SYS_PageInDocument' system variable when duplex pages are included in a document. |
| SYS_SheetsUsedbyPaperTypes | The number of sheets of each paper type used during an engine run |
| SYS_SheetTotalInDocument | The total number of sheets included in the document The value of the 'SYS_SheetTotalInDocument' system variable might vary from the value of the 'SYS_PageTotalInDocument' system variable when duplex pages are included in a document. |
| SYS_SortIndexDocument | A 15-byte variable used in sort index files to select documents to include and exclude for certain customers The 'SYS_SortIndexDocument' system variable is used only in post-sort processing. |
| SYS_SubDocInDocument | The name of the current document in the customer |
| SYS_StyleSheet | The name of the currently selected dynamic style sheet |
| SYS_TotalSubDocsInDocument | The total number of documents included in the customer |

3.5.2 Counting and Numbering Related System Variables

The system variables in the following table are related to counting a type of object in an application or numbering an object as it is processed. They provide information such as the number of rows in a table, the total number of pages in an engine run, or a specific page number.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Counting and numbering related system variables

| Variable name | Description |
|---------------------------|--|
| SYS_BannerCurrenPageInEnv | The sequence number of the current page within the envelope |
| SYS_BannerCurrEnvelope | The sequence number of the current banner page within the envelope |
| SYS_BannerEnvelopeInBreak | The sequence number of the current envelope within the convenience break |

Counting and numbering related system variables, continued

| Variable name | Description |
|-----------------------------|--|
| SYS_BannerEnvelopesInQueue | The total number of envelopes created for this entire queue. |
| SYS_BannerTotalEnvelope | The total number of envelopes included in the customer |
| SYS_BannerTotalPagesInEnv | The total number of pages included in the envelope |
| SYS_BankPagesInBreak | The total number of blank pages created for this break in the queue. |
| SYS_BankPagesInQueue | The total number of blank pages created using the output queue |
| SYS_BankPageTotalInRun | The total number of blank pages included in an engine run |
| SYS_BreaksInQueue | The total number of convenience breaks created using the current output queue |
| SYS_BundleCustomer1 | The sequence number of the current customer within the first tier of bundling |
| SYS_BundleCustomer2 | The sequence number of the current customer within the second tier of bundling |
| SYS_BundleCustomer3 | The sequence number of the current customer within the third tier of bundling |
| SYS_BundleCustomer4 | The sequence number of the current customer within the fourth tier of bundling |
| SYS_BundleCustomer5 | The sequence number of the current customer within the fifth tier of bundling |
| SYS_BundleCustomerTotal1 | The total number of customers included in the first tier of bundling |
| SYS_BundleCustomerTotal2 | The total number of customers included in the second tier of bundling |
| SYS_BundleCustomerTotal3 | The total number of customers included in the third tier of bundling |
| SYS_BundleCustomerTotal4 | The total number of customers included in the fourth tier of bundling |
| SYS_BundleCustomerTotal5 | The total number of customers included in the fifth tier of bundling |
| SYS_BundleInBreak1 | The sequence number of the current bundle within the convenience break for the first tier of bundling |
| SYS_BundleInBreak2 | The sequence number of the current bundle within the convenience break for the second tier of bundling |
| SYS_BundleInBreak3 | The sequence number of the current bundle within the convenience break for the third tier of bundling |
| SYS_BundleInBreak4 | The sequence number of the current bundle within the convenience break for the fourth tier of bundling |
| SYS_BundleInBreak5 | The sequence number of the current bundle within the convenience break for the fifth tier of bundling |

Counting and numbering related system variables, continued

| Variable name | Description |
|-----------------------|---|
| SYS_BundleInQueue1 | The sequence number of the current bundle within the output queue for the first tier of bundling |
| SYS_BundleInQueue2 | The sequence number of the current bundle within the output queue for the second tier of bundling |
| SYS_BundleInQueue3 | The sequence number of the current bundle within the output queue for the third tier of bundling |
| SYS_BundleInQueue4 | The sequence number of the current bundle within the output queue for the fourth tier of bundling |
| SYS_BundleInQueue5 | The sequence number of the current bundle within the output queue for the fifth tier of bundling |
| SYS_BundlePage1 | The sequence number of the current page within the first tier of bundling |
| SYS_BundlePage2 | The sequence number of the current page within the second tier of bundling |
| SYS_BundlePage3 | The sequence number of the current page within the third tier of bundling |
| SYS_BundlePage4 | The sequence number of the current page within the fourth tier of bundling |
| SYS_BundlePage5 | The sequence number of the current page within the fifth tier of bundling |
| SYS_BundlePageTotal1 | The total number of pages included in the first tier of bundling |
| SYS_BundlePageTotal2 | The total number of pages included in the second tier of bundling |
| SYS_BundlePageTotal3 | The total number of pages included in the third tier of bundling |
| SYS_BundlePageTotal4 | The sequence number of the current sheet within the fourth tier of bundling |
| SYS_BundlePageTotal5 | The total number of pages included in the fifth tier of bundling |
| SYS_BundleSheet1 | The sequence number of the current sheet within the first tier of bundling |
| SYS_BundleSheet2 | The sequence number of the current sheet within the second tier of bundling |
| SYS_BundleSheet3 | The sequence number of the current sheet within the third tier of bundling |
| SYS_BundleSheet4 | The sequence number of the current sheet within the fourth tier of bundling |
| SYS_BundleSheet5 | The sequence number of the current sheet within the fifth tier of bundling |
| SYS_BundleSheetTotal1 | The total number of sheets included in the first tier of bundling |
| SYS_BundleSheetTotal2 | The total number of sheets included in the second tier of bundling |

Counting and numbering related system variables, continued

| Variable name | Description |
|-----------------------------|---|
| SYS_BundleSheetTotal3 | The total number of sheets included the third tier of bundling |
| SYS_BundleSheetTotal4 | The total number of sheets included the fourth tier of bundling |
| SYS_BundleSheetTotal5 | The total number of sheets included the fifth tier of bundling |
| SYS_BytelnBreak | The total number of bytes written to the current convenience break of the output queue |
| SYS_BytelnQueue | The total number of bytes written to the current output queue |
| SYS_CampaignFirstPage | The printed page number of the first page on which the campaign appears <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Tip: You can use the 'SYS_CampaignFirstPage' system variable with teaser messages to reference the location of the campaign it is introducing </div> |
| SYS_CampaignsQualified | The total number of campaigns qualified for the customer |
| SYS_CampaignsSent | The total number of campaigns that were sent to the customer |
| SYS_CustomerDocuments | The total number of documents included in the customer |
| SYS_CustomerInRun | The sequence number of the current customer within the engine run |
| SYS_DocFormattedValue | The formatted string used as the document or chapter number in a table of contents entry or on a page The value increments for each document in a customer and restarts at the beginning of a new customer or when a document specifies a restart. |
| SYS_DocPrintedValue | The value to use as the document or chapter number on the current page The number increments for each document in a customer and restarts at the beginning of a new customer or when a document specifies a restart. |
| SYS_DocumentTotalInRun | The total number of documents included in the engine run |
| SYS_MktgPagesAdded | The total number of pages added to the document because campaign messages required additional pages |
| SYS_MktgPagesAllowed | The maximum number of marketing pages that can be included in a customer mailing |
| SYS_MupPageInBreak | The sequence number of the current multiple-up page within the convenience break of the output queue |
| SYS_MupPageInDocument | The sequence number of the current multiple-up page within the document The 'SYS_MupPageInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_MupPageInQueue | The sequence number of the current multiple-up page within the output queue |

Counting and numbering related system variables, continued

| Variable name | Description |
|------------------------------------|--|
| SYS_MupPageInRun | The sequence number of the current multiple-up page within the engine run |
| SYS_MupPagePhysicalInDocument | The sequence number of the current physical multiple-up page within the customer The 'SYS_MupPagePhysicalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The number of physical pages might differ from counted pages because of the page counting method set on the application. |
| SYS_MupPageTotalInDocument | The total number of multiple-up pages included in the customer The 'SYS_MupPageTotalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The option you select for banner page counting in the application properties determines whether banner pages are counted. |
| SYS_MupPageTotalPhysicalInDocument | The total number of physical multiple-up pages included in the customer The 'SYS_MupPageTotalPhysicalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The number of physical pages might differ from counted pages because of the page counting method set on the application. |
| SYS_MupSheetInBreak | The sequence number of the current multiple-up sheet within the convenience break of the output queue |
| SYS_MupSheetInDocument | The sequence number of the current multiple-up sheet within the customer The 'SYS_MupSheetInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_MupSheetInQueue | The sequence number of the current multiple-up sheet within the output queue |
| SYS_MupSheetInRun | The sequence number of the current multiple-up sheet within the engine run |
| SYS_MupSheetTotalInDocument | The total number of sheets in the customer This count includes both multiple-up sheets and any additional pages that were created to accommodate design pages that could not be placed into MUP frames within the original design. Optionally, you can also select whether banner pages are included in the total sheet count. The 'SYS_MupSheetTotalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_NumBannersinQueue | The sequence number of the current banner page within the output queue |
| SYS_PageFormattedValue | The formatted string to use as the page number in a table of contents entry or on a page The page number might differ from page count because the page count might have been restarted or because of blank pages. Banner pages are not counted. |

Counting and numbering related system variables, continued

| Variable name | Description |
|---------------------------------|---|
| SYS_PageInBreak | The sequence number of the current page within the convenience break of the output queue Banner pages are counted. |
| SYS_PageInDocument | The sequence number of the current counted page within the document You can use the application properties to exclude some types of pages from the page count. Banner pages are not counted. The value of the 'SYS_PageInDocument' system variable might vary from the value of the 'SYS_SheetInDocument' system variable when duplex pages are included in a document. |
| SYS_PageInQueue | The total number of physical pages included in the current output queue Banner pages are counted. |
| SYS_PagePhysicalInDocument | The sequence number of the current physical page within the document The number of physical pages might differ from counted pages because of the page counting methods set on the application. |
| SYS_PagePrintedValue | The value to use as the page number of the current page The page number might differ from page count because the page count might have been restarted or because of blank pages. Banner pages do not have a printed value. |
| SYS_PagePrintEnd | The page number of the last physical page for page-level reprints The 'SYS_PagePrintEnd' system variable is used only with post-sort processing runs. |
| SYS_PagePrintStart | The page number of the first physical page for page-level reprints The 'SYS_PagePrintStart' system variable is used only with post-sort processing runs. |
| SYS_PageStart | The initial number of pages included in the document before pages are excluded for reasons such as rules or marketing page limits |
| SYS_PageTotalInDocument | The total number of counted pages included in the document The value of the 'SYS_PageTotalInDocument' system variable might vary from the value of the 'SYS_SheetTotalInDocument' system variable when duplex pages are included in a document. |
| SYS_PageTotalInRun | The total number of pages included in the run Banner pages are counted. |
| SYS_PageTotalPhysicalInDocument | The total number of physical pages included in the document The number of physical pages might differ from counted pages because of the page counting methods set on the application. |

Counting and numbering related system variables, continued

| Variable name | Description |
|----------------------------|---|
| SYS_PageTotalPrinted | The page number of the last page in the document The value of the 'SYS_PageTotalPrinted' system variable might vary from the value of the 'SYS_PageTotalPhysicalInDocument' system variable when page counts reset at the start of a document. |
| SYS_QueuestoCustomer | The total number of output queues for which the current customer qualified |
| SYS_RecordCount | The total number of records read from all of the customer driver files in an application |
| SYS_RecordCountDriver | The total number of records read from the current customer driver file |
| SYS_RecordInBreak | The total number of records written to the current convenience break of the output queue |
| SYS_RecordInQueue | The total number of records written to the current output queue |
| SYS_SheetInBreak | The sequence number of the current sheet within the convenience break of the output queue |
| SYS_SheetInDocument | The sequence number of the current sheet within the document The value of the 'SYS_SheetInDocument' system variable might vary from the value of the 'SYS_PageInDocument' system variable when duplex pages are included in a document. |
| SYS_SheetInQueue | The total number of sheets created using the output queue Banner pages are counted. |
| SYS_SheetsUsedbyPaperTypes | The number of sheets of each paper types used during an engine run |
| SYS_SheetTotalInDocument | The total number of sheets included in the document The value of the 'SYS_SheetTotalInDocument' system variable might vary from the value of the 'SYS_PageTotalInDocument' system variable when duplex pages are included in a document. |
| SYS_TableColumn | The column count for an automated table column If the column repeats, the value of the 'SYS_TableColumn' system variable indicates the current count of the column, including those columns that were not composed. |
| SYS_TableRow | The sequence number of the current automated table row If the row repeats, the value of the 'SYS_TableRow' system variable indicates the current count of the row, including those rows that were not composed. |

Note: You can use the 'SYS_TableRow' system variable in formulas to compute transaction data on a row-by-row basis. The 'SYS_TableRow' system variable does not work for embedded tables.

Counting and numbering related system variables, continued

| Variable name | Description |
|----------------------------|--|
| SYS_TableRowOnPage | The sequence number of the current row on the page The value of the 'SYS_TableRowOnPage' system variable increments for each row included on the page that has the system variable. For example, if the variable is not included in the header, then the header is not counted. For an automated row that is included multiple times, the system variable increments each time the row is included. You cannot use the 'SYS_TableRowOnPage' system variable in row rules. |
| SYS_TableRowPage | The total number of rows on the current page The value of the 'SYS_TableRowPage' system variable does not include headers, footers, or rows that were excluded because of rules. |
| SYS_TableRowTotal | The total number of rows on all of the pages The value of the 'SYS_TableRowTotal' system variable does not include headers, footers, or rows that were excluded because of rules. |
| SYS_TotalSubDocsInDocument | The total number of documents included in a customer |

3.5.3 Customer Related System Variables

The system variables in the following table are related to information about a customer. They provide information such as the number of documents sent to a customer, where a customer lives, or the default language for all of the customers in an engine run.

In the table below, if a system variable name is in **bold**, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Customer related system variables

| Variable name | Description |
|--------------------------|--|
| SYS_BundleCustomer1 | The sequence number of the current customer within the first tier of bundling |
| SYS_BundleCustomer2 | The sequence number of the current customer within the second tier of bundling |
| SYS_BundleCustomer3 | The sequence number of the current customer within the third tier of bundling |
| SYS_BundleCustomer4 | The sequence number of the current customer within the fourth tier of bundling |
| SYS_BundleCustomer5 | The sequence number of the current customer within the fifth tier of bundling |
| SYS_BundleCustomerTotal1 | The total number of customers included in the first tier of bundling |
| SYS_BundleCustomerTotal2 | The total number of customers included in the second tier of bundling |

Customer related system variables, continued

| Variable name | Description |
|---------------------------|---|
| SYS_BundleCustomerTotal3 | The total number of customers included in the third tier of bundling |
| SYS_BundleCustomerTotal4 | The total number of customers included in the fourth tier of bundling |
| SYS_BundleCustomerTotal5 | The total number of customers included in the fifth tier of bundling |
| SYS_CustInvalidDataLevel | An invalid data level for the current customer, set when the engine reads or computes a variable and detects invalid data. The level is based on the action set in the variable's validation. The flag indicates the maximum value read for a customer: <ul style="list-style-type: none">• 0 = No invalid data• 1 = Invalid data, continue, or set to default• 2 = Invalid data, error• 10 = Invalid data in this document. Skips all other documents for the customer. <p>Tip: You can also set the 'SYS_CustInvalidDataLevel' system variable using the MSGCHANGE engine switch or the Message function.</p> <p>For more information about the MSGCHANGE engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>For more information about the Message function, see <i>Using Logic to Drive an Application</i> in the Exstream Design and Production documentation.</p> |
| SYS_CustomerBegByte | The integer byte offset in the convenience break of the output queue file for the beginning of the current customer |
| SYS_CustomerBegRecord | The integer record position in the convenience break of the output queue file for the beginning of the current customer |
| SYS_CustomerChecksum | The checksum value, which is used to validate an object has been sent, of the text and images in the document or customer <p>For more information about checksums, see <i>Creating Output</i> in the Exstream Design and Production documentation.</p> |
| SYS_CustomerDocuments | The total number of documents included in the customer |
| SYS_CustomerEffectiveDate | The date used to select the correct version of an object for a customer based on when the object is effective <p>For more information about effectiveness, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> |
| SYS_CustomerInRun | The sequence number of the current customer within the engine run |
| SYS_CustomerJurisdiction | The name of the jurisdiction associated with the current customer |

Customer related system variables, continued

| Variable name | Description |
|-----------------------------|--|
| SYS_CustomerQueues | The list of output queue names that is enabled for a customer. Keep in mind that when queue rules are processed during the engine run, output queues that do not appear in this list will be disabled. |
| SYS_DocumentNames | The names of the documents included in the customer |
| SYS_LanguageCustomer | The language of the current customer, which is used to identify the values to use when dynamically selecting language-based objects and settings |
| SYS_LanguageDefault | The default language for all of the customers within the engine run |
| SYS_LocaleCustomer | The locale for the current customer, which is used to identify the variables that dynamically select the appropriate locale-based objects and settings |
| SYS_LocaleDefault | The default locale for all of the customers within the engine run |
| SYS_TotalSubDocsInDocument | The total number of documents included in a customer |

3.5.4 Data Related System Variables

The system variables in the following table are related to data files. They provide information such as the name of a parent section in XML data, the name of the record that begins a data section, or the number of records read in the current customer driver file.

For more information about data files, see [“Setting Up Data Files” on page 9](#).

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Data related system variables

| Variable name | Description |
|---------------------------------|--|
| SYS_CustInvalidDataLevel | <p>An invalid data level for the current customer, set when the engine reads or computes a variable and detects invalid data. The level is based on the action set in the variable's validation. The flag indicates the maximum value read for a customer:</p> <ul style="list-style-type: none">• 0 = No invalid data• 1 = Invalid data, continue, or set to default• 2 = Invalid data, error• 10 = Invalid data in this document. Skips all other documents for the customer. <p>Tip: You can also set the 'SYS_CustInvalidDataLevel' system variable using the MSGCHANGE engine switch or the Message function.</p> <p>For more information about the MSGCHANGE engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>For more information about the Message function, see <i>Using Logic to Drive an Application</i> in the Exstream Design and Production documentation.</p> |
| SYS_DriverFileOffsetBytes | <p>The offset in the customer driver file before any data is read for the current customer</p> <p>The value of the 'SYS_DriverFileOffsetBytes' system variable is returned as a floating point number to allow offsets larger than 4GB. You should use the Fixed Decimal formatting. If you use General Number and the Data Type is Floating, you receive four decimal places.</p> |
| SYS_DriverFileOffsetBytesEnd | <p>The byte offset in the customer driver file after the data was read for the current customer</p> <p>The value of the 'SYS_DriverFileOffsetBytesEnd' system variable is returned as a floating point number to allow offsets larger than 4GB. You should use the Fixed Decimal formatting. If you use General Number and the Data Type is Floating, you receive four decimal places.</p> <p>The Reset time of the 'SYS_DriverFileOffsetBytesEnd' system variable is automatically set to Before each customer.</p> |
| SYS_DriverFileOffsetRecs | <p>The record position in the customer driver file before any data was read for the current customer</p> |
| SYS_DriverFileOffsetRecsEnd | <p>The record position in the customer driver file after data was read for the current customer</p> <p>The value is returned as an integer. The Reset time of the 'SYS_DriverFileOffsetRecsEnd' system variable is automatically set to Before each customer.</p> |
| SYS_ParentSectionName | <p>For hierarchical XML data, the name of the parent data section of the current data section</p> |
| SYS_ParentSectionInstanceNumber | <p>For hierarchical XML data, the instance number of the parent data section of the current active data section</p> <p>The 'SYS_ParentSectionInstanceNumber' system variable applies only to sections within data sections having the same name.</p> |
| SYS_RecordCount | <p>The total number of records read from all of the customer driver files in an application</p> |
| SYS_RecordCountDriver | <p>The total number of records read from the current customer driver file</p> |

Data related system variables, continued

| Variable name | Description |
|-------------------------------|---|
| SYS_SectionIndex | The absolute index of the current data section within all instances of section data, regardless of data section name |
| SYS_SectionLevelNumber | For hierarchical XML data, the level within the hierarchy of the current data section |
| SYS_SectionName | The name of the record that caused the current data section to begin If no sections are being processed, the value is empty. |
| SYS_SectionNameInstanceNumber | The instance number of the current data section within all data sections having the same name |
| SYS_SortIndex | A 20-byte variable used in sort index files for the indexing information. The 'SYS_SortIndex' system variable is used only in post-sort processing. |
| SYS_SortIndexDocument | A 15-byte variable used in sort index files to select documents to include and exclude for certain customers The 'SYS_SortIndexDocument' system variable is used only in post-sort processing. |
| SYS_ValidationMessage | An optional custom error message returned from a variable validation function |
| SYS_ValidationVariable | The variable used to access the data to be validated within a variable validation function |

3.5.5 Date Related System Variables

The system variables in the following table are related to dates. They provide information such as the date a package was created or the date on which the engine is run.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Date related system variables

| Variable name | Description |
|----------------------------------|---|
| SYS_CustomerEffectiveDate | The date used to select the correct version of an object based on when the object is effective For more information about effectiveness, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation. |
| SYS_DateAsOf | The day on which the package was created The date might vary from the calendar date. |
| SYS_DateCurrent | The day on which the engine is run to create documents |

3.5.6 Design Related System Variables

The system variables in the following table are related to the design objects placed on a page. They provide information such as the number of table rows, or the language of the customer.

For more information about designing a page, see *Designing Customer Communications* in the Exstream Design and Production documentation.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Design related system variables

| Variable name | Description |
|---------------------------------|--|
| SYS_CustomerJurisdiction | The name of the jurisdiction associated with the current customer |
| SYS_LanguageCustomer | The language of the current customer, which is used to identify the values to use when dynamically selecting language based objects and settings. |
| SYS_LanguageDefault | The default language for all of the customers within one engine run |
| SYS_LocaleCustomer | The locale for the current customer, which is used to identify the variables that dynamically select the appropriate locale-based objects and settings |
| SYS_LocaleDefault | The default locale for all of the customers within the engine run. |
| SYS_TableColumn | The column count for an automated table column If the column repeats, the value of the 'SYS_TableColumn' system variable indicates the current count of the column, including those columns that were not composed. |
| SYS_TableRow | The sequence number of the current automated table row If the row repeats, the value of the 'SYS_TableRow' system variable indicates the current count of the row, including those rows that were not composed. <div style="border: 1px solid #ccc; padding: 5px;"><p>Note: You can use the 'SYS_TableRow' system variable in formulas to compute transaction data on a row-by-row basis. The 'SYS_TableRow' system variable does not work for embedded tables.</p></div> |
| SYS_TableRowOnPage | The sequence number of the current row on the page The value of the 'SYS_TableRowOnPage' system variable increases by one for each row included on the page that has the system variable. For example, if the variable is not included in the header, then the header is not counted. For an automated row that is included multiple times, the system variable increments each time the row is included. You cannot use the 'SYS_TableRowOnPage' system variable in row rules. |

Design related system variables, continued

| Variable name | Description |
|-------------------|--|
| SYS_TableRowPage | The total number of rows on the current page The value of the 'SYS_TableRowPage' system variable does not include headers, footers, or rows that were excluded because of rules. |
| SYS_TableRowTotal | The total number of rows on all of the pages The value of the 'SYS_TableRowTotal' system variable does not include headers, footers, or rows that were excluded because of rules. |

3.5.7 High-Volume Output Processing Related System Variables

The system variables in the following table are related to output processes such as sorting and bundling, and to objects such as multiple up pages and inserters. They provide information such as the number of multiple-up pages in a convenience break, the name of the output device associated with an output queue, or whether you use a rule before or after sorting.

For more information about output processing, see *Creating Output* in the Exstream Design and Production documentation.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

High-volume output processing related system variables

| Variable name | Description |
|---------------------------|--|
| SYS_BannerCurrenPageInEnv | The sequence number of the current page within the envelope |
| SYS_BannerCurrEnvelope | The sequence number of the current banner page within the envelope |
| SYS_BannerEnvelopeInBreak | The sequence number of the current envelope within the convenience break |
| SYS_BannerEnvelopeInQueue | The total number of envelopes created for this entire queue. |
| SYS_BannerTotalEnvelope | The total number of envelopes included in the customer |
| SYS_BannerTotalPagesInEnv | The total number of pages included in the envelope |
| SYS_BlankPageInBreak | The total number of blank pages created for this break in the queue. |
| SYS_BlankPageInQueue | The total number of blank pages created using the output queue |

High-volume output processing related system variables, continued

| Variable name | Description |
|--------------------------|--|
| SYS_BreaksInQueue | The total number of convenience breaks created using the current output queue |
| SYS_BundleCustomer1 | The sequence number of the current customer within the first tier of bundling |
| SYS_BundleCustomer2 | The sequence number of the current customer within the second tier of bundling |
| SYS_BundleCustomer3 | The sequence number of the current customer within the third tier of bundling |
| SYS_BundleCustomer4 | The sequence number of the current customer within the fourth tier of bundling |
| SYS_BundleCustomer5 | The sequence number of the current customer within the fifth tier of bundling |
| SYS_BundleCustomerTotal1 | The total number of customers included in the first tier of bundling |
| SYS_BundleCustomerTotal2 | The total number of customers included in the second tier of bundling |
| SYS_BundleCustomerTotal3 | The total number of customers included in the third tier of bundling |
| SYS_BundleCustomerTotal4 | The total number of customers included in the fourth tier of bundling |
| SYS_BundleCustomerTotal5 | The total number of customers included in the fifth tier of bundling |
| SYS_BundleInBreak1 | The sequence number of the current bundle within the convenience break for the first tier of bundling |
| SYS_BundleInBreak2 | The sequence number of the current bundle within the convenience break for the second tier of bundling |
| SYS_BundleInBreak3 | The sequence number of the current bundle within the convenience break for the third tier of bundling |
| SYS_BundleInBreak4 | The sequence number of the current bundle within the convenience break for the fourth tier of bundling |
| SYS_BundleInBreak5 | The sequence number of the current bundle within the convenience break for the fifth tier of bundling |
| SYS_BundleInQueue1 | The sequence number of the current bundle within the output queue for the first tier of bundling |
| SYS_BundleInQueue2 | The sequence number of the current bundle within the output queue for the second tier of bundling |
| SYS_BundleInQueue3 | The sequence number of the current bundle within the output queue for the third tier of bundling |
| SYS_BundleInQueue4 | The sequence number of the current bundle within the output queue for the fourth tier of bundling |

High-volume output processing related system variables, continued

| Variable name | Description |
|-----------------------|--|
| SYS_BundleInQueue5 | The sequence number of the current bundle within the output queue for the fifth tier of bundling |
| SYS_BundlePage1 | The sequence number of the current page within the first tier of bundling |
| SYS_BundlePage2 | The sequence number of the current page within the second tier of bundling |
| SYS_BundlePage3 | The sequence number of the current page within the third tier of bundling |
| SYS_BundlePage4 | The sequence number of the current page within the fourth tier of bundling |
| SYS_BundlePage5 | The sequence number of the current page within the fifth tier of bundling |
| SYS_BundlePageTotal1 | The total number of pages included in the first tier of bundling |
| SYS_BundlePageTotal2 | The total number of pages included in the second tier of bundling |
| SYS_BundlePageTotal3 | The total number of pages included in the third tier of bundling |
| SYS_BundlePageTotal4 | The total number of pages included in the fourth tier of bundling |
| SYS_BundlePageTotal5 | The total number of pages included in the fifth tier of bundling |
| SYS_BundleSheet1 | The sequence number of the current sheet within the first tier of bundling |
| SYS_BundleSheet2 | The sequence number of the current sheet within the second tier of bundling |
| SYS_BundleSheet3 | The sequence number of the current sheet within the third tier of bundling |
| SYS_BundleSheet4 | The sequence number of the current sheet within the fourth tier of bundling |
| SYS_BundleSheet5 | The sequence number of the current sheet within the fifth tier of bundling |
| SYS_BundleSheetTotal1 | The total number of sheets included in the first tier of bundling |
| SYS_BundleSheetTotal2 | The total number of sheets included the second tier of bundling |
| SYS_BundleSheetTotal3 | The total number of sheets included the third tier of bundling |
| SYS_BundleSheetTotal4 | The total number of sheets included the fourth tier of bundling |
| SYS_BundleSheetTotal5 | The total number of sheets included the fifth tier of bundling |
| SYS_BytelnBreak | The total number of bytes written to the current break of the current output queue |
| SYS_BytelnQueue | The total number of bytes written to the current output queue |

High-volume output processing related system variables, continued

| Variable name | Description |
|--|--|
| SYS_CustomerBegByte | The integer byte offset in the convenience break of the output queue for the beginning of the current customer |
| SYS_CustomerBegRecord | The integer record position in the convenience break of the output queue for the beginning of the current customer |
| SYS_DocumentInBreak | The sequence number of the current document within the convenience break of the output queue |
| SYS_DocumentInQueue | The sequence number of the current customer within the output queue |
| SYS_InserterBinPriorities | An array that lists the priority of the messages in each inserter bin for the current customer in the output queue |
| <p>Tip: You can use the Count function to determine the number of output queues used.</p> | |
| SYS_InserterBins | An array of Boolean values that specifies which inserter bins that have been selected for the current customer in the output queue |
| <p>Tip: You can use the Count function to determine the number of output queues used.</p> | |
| SYS_InserterBinString | A string containing a 1 for each inserter bin selected and a 0 for each inserter bin not selected for the current customer |
| <p>Tip: You can use the Count function to determine the number of output queues used.</p> | |
| SYS_InserterInQueue | The name you have specified for the current inserter bin in the output queue |
| SYS_MupPageInBreak | The sequence number of the current multiple-up page within the convenience break of the output queue |
| SYS_MupPageInDocument | The sequence number of the current multiple-up page within the document The 'SYS_MupPageInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_MupPageInQueue | The sequence number of the current multiple-up page within the output queue |
| SYS_MupPageInRun | The sequence number of the current multiple-up page within the engine run |
| SYS_MupPagePhysicalInDocument | The sequence number of the current physical multiple-up page within the customer The 'SYS_MupPagePhysicalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The number of physical pages might differ from counted pages because of the page counting method set on the application. |

High-volume output processing related system variables, continued

| Variable name | Description |
|------------------------------------|--|
| SYS_MupPageTotalInDocument | The total number of multiple-up pages included in the customer The 'SYS_MupPageTotalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The option you select for banner page counting in the application properties determines whether banner pages are counted. |
| SYS_MupPageTotalPhysicalInDocument | The total number of physical multiple-up pages included in the customer The 'SYS_MupPageTotalPhysicalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. The number of physical pages might differ from counted pages because of the page counting method set on the application. |
| SYS_MupSheetInBreak | The sequence number of the current multiple-up sheet within the convenience break of the output queue |
| SYS_MupSheetInDocument | The sequence number of the current multiple-up sheet within the customer The 'SYS_MupSheetInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_MupSheetInQueue | The sequence number of the current multiple-up sheet within the output queue |
| SYS_MupSheetInRun | The sequence number of the current multiple-up sheet within the engine run |
| SYS_MupSheetTotalInDocument | The total number of sheets in the customer This count includes both multiple-up sheets and any additional pages that were created to accommodate design pages that could not be placed into MUP frames within the original design. Optionally, you can also select whether banner pages are included in the total sheet count. The 'SYS_MupSheetTotalInDocument' system variable is not valid if you combine multiple customers on a multiple-up page. |
| SYS_NumBannersinQueue | The sequence number of the current banner page within the output queue |
| SYS_PageInBreak | The sequence number of the current page within the convenience break of the output queue Banner pages are counted. |
| SYS_PageInQueue | The total number of physical pages included in the current output queue. Banner pages are counted. |
| SYS_PagePrintEnd | The page number of the last physical page for page-level reprints The 'SYS_PagePrintEnd' system variable is used only with post-sort processing runs. |
| SYS_PagePrintStart | The page number of the first physical page for page-level reprints The 'SYS_PagePrintStart' system variable is used only with post-sort processing runs. |
| SYS_PDLInQueue | The type of output device that is associated with the current output queue |

High-volume output processing related system variables, continued

| Variable name | Description |
|--------------------------|---|
| SYS_PostSort | A Boolean value that returns T in post-sort and F in pre-sort processing |
| SYS_PrinterInQueue | The name you have specified for the output device that is associated with the current output queue |
| SYS_QueueCurrent | The name you have specified for the current output queue |
| SYS_QueueFileName | The system file name and location of the current output queue The system file name (C : \EXSTREAM\OUTPUT . PS, for example) is the queue name used for banner pages and archives. |
| SYS_QueueReportFileName | The current output queue report file name |
| SYS_QueuesToCustomer | The total number of output queues for which the current customer qualified |
| SYS_RecordInBreak | The total number of records written to the current convenience break of the output queue |
| SYS_RecordInQueue | The total number of records written to the current output queue |
| SYS_SheetInBreak | The sequence number of the current sheet within the convenience break of the output queue |
| SYS_SheetInQueue | The sequence number of the current sheet within the output queue Banner pages are counted. |
| SYS_SortIndex | A 20-byte variable used in sort index files for the indexing information. The 'SYS_SortIndex' system variable is used only in post-sort processing. |
| SYS_SortIndexDocument | A 15-byte variable used in sort index files to select documents to include and exclude for certain customers The 'SYS_SortIndexDocument' system variable is used only in post-sort processing. |
| SYS_SubDocumentInBreak | The sequence number of the current document within the convenience break of the output queue |
| SYS_SubDocumentInQueue | The sequence number of the current document within the output queue |

3.5.8 Marketing Related System Variables

The system variables in the following table are related to campaigns, messages, and tracking. They provide information such as the number of marketing pages that can be included, the names of all the messages in an engine run, or the number of campaigns sent to a customer.

For more information about messages, campaigns, and tracking, see *Managing Marketing Messages* in the Exstream Design and Production documentation.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Marketing related system variables

| Variable name | Description |
|----------------------------------|--|
| SYS_CampaignFirstPage | The printed page number of the first page on which the campaign appears Tip: You can use the 'SYS_CampaignFirstPage' system variable with teaser messages to reference the location of the campaign it is introducing. |
| SYS_CampaignPriority | The priority of the campaign if it is set by a campaign priority formula |
| SYS_CampaignsQualified | The total number of campaigns qualified for the customer |
| SYS_CampaignsSent | The total number of campaigns that were sent to the customer |
| SYS_InserterBinPriorities | An array that lists the priority of the messages in each inserter bin for the current customer in the output queue Tip: You can use the Count function to determine the number of output queues used. |
| SYS_MktgPagesAdded | The total number of pages added to the document because campaign messages required additional pages |
| SYS_MktgPagesAllowed | The maximum number of marketing pages that can be included in a customer mailing |
| SYS_MsgContents | An array that lists the contents of all the text messages in an engine run |
| SYS_MsgContentsFormatted | An array that lists the contents, including formatting, of all the text messages in an engine run |
| SYS_MsgIdentifier | An array that lists the message identifiers of all the messages in an engine run |
| SYS_MsgName | An array that lists the names of all the messages in an engine run |
| SYS_MsgTypes | An array that lists the message types of all the messages in an engine run |
| SYS_VirtMsg_Contents | The unformatted text contents of the current virtual text message within an application The contents of the 'SYS_VirtMsg_Contents' system variable change as each message is processed. |
| SYS_VirtMsg_Identifier | The name you have specified in the message properties as the identifier for a virtual message |
| SYS_VirtMsg_Name | The name of the virtual messages selected for an application |
| SYS_VirtMsg_Priority | The priority of the virtual messages selected for an application |

3.5.9 Production Related System Variables

The system variables in the following table are related to packaging and running the engine. They provide information such as the contents of all the messages in an engine run, the date a package was created, and the weight of a document after it has been packaged.

For more information about production process, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Production related system variables

| Variable name | Description |
|-----------------------------|--|
| SYS_BankPageTotalInRun | The total number of blank pages included in an engine run |
| SYS_CustomerChecksum | The checksum value, which is used to validate an object has been sent, of the text and images in the document or customer For more information about checksums, see <i>Creating Output</i> in the Exstream Design and Production documentation. |
| SYS_CustomerInRun | The sequence number of the current customer within the engine run |
| SYS_DateAsOf | The day on which the package was created The date might vary from the calendar date. |
| SYS_DateCurrent | The day on which the engine is run to create documents |
| SYS_DocumentChecksums | An array that lists the checksum values, which are used to validate an object has been sent, for all of the documents included in the current customer |
| SYS_DocumentTotalInRun | The total number of documents included in the engine run |
| SYS_LanguageCustomer | The default language for all of the customers within the engine run |
| SYS_LanguageDefault | The default locale for all of the customers within the engine run |
| SYS_MsgContents | An array that lists the contents of all the text messages in an engine run |
| SYS_MsgContentsFormatted | An array that lists the contents, including formatting, of all the text messages in an engine run |
| SYS_MsgIdentifier | An array that lists the message identifiers of all the messages in an engine run |
| SYS_MsgName | An array that lists the names of all the messages in an engine run |

Production related system variables, continued

| Variable name | Description |
|----------------------------|--|
| SYS_MsgTypes | An array that lists the message types of all the messages in an engine run |
| SYS_MupPageInRun | The sequence number of the current multiple-up page within the engine run |
| SYS_MupSheetInRun | The sequence number of the current multiple-up sheet within the engine run |
| SYS_PageTotalInRun | The total number of pages included in the run Banner pages are counted. |
| SYS_SheetsUsedbyPaperTypes | The number of sheets of each paper types used during an engine run |
| SYS_SheetTotalInRun | The total number of sheets included in the engine run Banner pages are not counted. |
| SYS_VolSerInQueue | The Volume Serial of a z/OS data set or tape |
| SYS_WeightGrams | The weight of the current document, in grams, after packaging has been completed |
| SYS_WeightOunces | The weight of the current document, in ounces, after packaging has been completed |
| SYS_WeightStartGrams | The initial weight of the current document, in grams, before packaging has been completed |
| SYS_WeightStartOunces | The initial weight of the current document, in ounces, before packaging has been completed |

3.5.10 Real-time and SOAP-Related System Variables

The system variables in the following table are related to real-time and dynamic access to data. They provide information such as a SOAPAction HTTP header, the Web address used in a web service call, or a string to append to output when using a DDA routine.

The **-SYS_Soap***- variables are not required to be mapped. These values are reset during use, so they cannot be used after invoking an additional web services file.

For more information about real-time and dynamic data access, see *Configuring Connectors* in the Exstream Design and Production documentation.

For more information about web service calls, see “[Setting Up Data Files to Communicate with a Web Service](#)” on page 272.

In this section, if a system variable name is in bold, you can calculate its value using a rule or formula.

For more information about creating a rule or a formula, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

Real-time and SOAP-related system variables

| Variable name | Description |
|------------------------------|--|
| SYS_CorrelationID | A string that contains the current Dynamic Data Access (DDA) correlation ID |
| SYS_DDAOutputUserData | A string that contains data to be appended to the output being sent to a DDA routine |
| SYS_SoapAction | The SOAPAction HTTP header used in the most recent web service call |
| SYS_SoapError | The text of the most recent system-generated error message received from a web service |
| SYS_SoapFault | An XML fragment, including the enclosing element, that contains the SOAP fault |
| SYS_SoapFaultActor | The text of the faultactor element (SOAP 1.1) in the SOAP fault |
| SYS_SoapFaultCode | The text of the faultcode element (SOAP 1.1) or the Value element within the Code element (SOAP 1.2) in the SOAP fault |
| SYS_SoapFaultDetail | The text of the detail element (SOAP 1.1) or the Detail element (SOAP 1.2) in the SOAP fault |
| SYS_SoapFaultNode | The text of the Node element (SOAP 1.2) in the SOAP fault |
| SYS_SoapFaultReason | The text of the first Text element within the Reason element (SOAP 1.2) in the SOAP fault |
| SYS_SoapFaultRole | The text of the Role element (SOAP 1.2) in the SOAP fault |
| SYS_SoapFaultString | The text of the faultstring element (SOAP 1.1) in the SOAP fault |
| SYS_SoapFaultSubcode | The text of the Value element within the first Subcode element within the Code element (SOAP 1.2) in the SOAP fault |
| SYS_SoapURL | The Web address for the target server used in the most recent web service call |

Chapter 4: Data File Mapping

In order for the engine to use your data to create customized documents, you must complete a process called data mapping. When you map a data file, you associate a variable with a data area. This process creates data maps that the engine uses to find data. The engine then uses the data to populate variables and customize documents. The variable mapped to the data area defines the type of data found in the area. During mapping, you also define how to separate different types of data from each other and the use and format of the data area. For example, suppose the data in the area is a date. When the engine populates the variable during production, the date in the specified area replaces the variable.

Exstream Design and Production provides tools that allow you to easily map data files. There are two methods that you can use to map a data file: automapping and manual mapping. Both methods are completed in the Edit Panel in Design Manager and are usually completed only one time per data file. Automapping lets you automatically find and map areas in an entire data file, while manual mapping lets you define individual data areas. In many cases, you use the two mapping methods together to map a data file then make adjustments.

In most cases, after you initially map a data file and put an application into production, no subsequent changes are required. When you define a data file object, you define the location of a test data source and a production data source. You map the test data source. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. The test data source is not the data source the engine uses to produce applications, however. During production, the engine uses the data maps, along with the production data source, to read and write customer data. The data in the production data source can change. For example, if you create monthly account statements, you can update the production data source every month. Even if you update the production data source, you are not required to map the data file again, which means you can update customer information without updating the data file object or the data map.

For more information about defining the test and production versions of a data file, see [“Selecting the Customer Driver File Data Source” on page 27](#).

In addition, depending on how your organization uses and creates data files, you can create a single data file that you can use in multiple applications or you can use the same data sources in two different data files. Before mapping a data file, consider the data each application uses. You are not required to map variables to all of the data areas in a data file, which means even if you use the same data sources, you can create two different data maps. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory required, map only those data areas you want to use to populate variables. For example, suppose that you have a correspondence application and a statement application. If the two applications require similar data, you can create a single data file that you include in both applications. If the correspondence application requires only a portion of the data required for the statement application, you can use the same data sources, but map only the data used by the correspondence application.

To map a data file, complete the following tasks as needed:

- “[Setting Up the Data Mapping View](#)” below
- “[Automapping a Data File](#)” on page 135
- “[Manually Mapping a Data File](#)” on page 146
- “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151
- “[Modifying a Data Map](#)” on page 157
- “[Testing a Data Map](#)” on page 159
- “[Searching a Mapped Data File](#)” on page 160

4.1 Setting Up the Data Mapping View

Before or during mapping, you can change what you see in both the Edit Panel and the Property Panel. For example, you can change the data mapping view to accommodate mapping preferences or to accommodate the size or format of the data files.

To set up the data mapping view, complete the following tasks as needed:

- “[Selecting the Default Data Mapping View of the Property Panel](#)” below
- “[Limiting the Data Records That Appear in the Edit Panel](#)” on page 133
- “[Changing the Text Encoding in the Edit Panel](#)” on page 133
- “[Changing the Font in the Edit Panel](#)” on page 134
- “[Viewing the Hexadecimal Value of the Text](#)” on page 134

4.1.1 Selecting the Default Data Mapping View of the Property Panel

Although you must map a data file in the Edit Panel, you can view additional mapping information in the Property Panel. You can select a default view to make the data mapping process faster or to help you verify the data map of the data file.

To select the default data mapping view of the Property Panel:

1. From the **Tools** menu, select **Options**.
The **Options** dialog box opens.
2. To specify the default data mapping view of the Property Panel, select one of the following

options from the **Top panel initialization** list:

- **Do not show variables**—No extra information is opened in the Property Panel.
- **Show data dictionary**—Open all of the available variables in the Property Panel.
- **Show data layouts**—In the Property Panel, open details about the variables mapped to the data file.

3. Click **OK**.

The **Options** dialog box closes.

Using the Data Layout View

The data layout view includes all of the variables and records mapped to the data file. By default, the variables and records are listed in the order of occurrence. A multicolored record separator, associated with the record type indicator, differentiates one record layout from another. The variables mapped to the record layout are located between the separators.

For more information about mapping record type indicators, see “[Mapping a Variable to Sample Data in a Data Area](#)” on page 146.

Sample data layout view in the Property Panel

| Layouts for Loan_Details | | |
|--------------------------|----------------------------|----------|
| Name | Description | Type |
| MemberID | New Customer (5 Areas) | Record |
| LoanAmt | | Integer |
| LoanPaymentAmt | | Currency |
| LoanTerm | | Currency |
| LoanInterestRate | | Integer |
| MemberNameFirst | New Section: 110 (2 Areas) | Float |
| MemberNameLast | | Record |
| MemberNameFirst | | String |
| MemberNameLast | | String |

1 Variable

2 Record separator

To use the data layout view:

1. To open the data layout view, do one of the following :

| To | Do this |
|---|--|
| Open the data layout view as your default view each time you open a data file in the Edit Panel | <ol style="list-style-type: none">a. From the Tools menu, select Options. The Options dialog box opens.b. From the Top panel initialization list select Show data layouts.c. Click OK. The Options dialog box closes. |
| Open the data layout view if it is not already available in the Property Panel | Click  on the Data Mapping toolbar. |

2. To update indicator and section options, do the following:

- a. Double-click a record separator.

The **Record Properties** dialog box opens.

- b. Update options on the dialog box.

- c. Click **OK**.

The **Record Properties** dialog box closes.

3. To find a variable in the data file, highlight a variable name in the Property Panel.

A black line appears around the associated data area in the Edit Panel.

4. To update the use and format of the data area, do the following:

- a. Double-click a variable name.

The **Data Area Properties** dialog box opens.

- b. Update options on the dialog box.

- c. Click **OK**.

The **Data Area Properties** dialog box closes.

For more information about data area uses and formats, see ["Using a Data Area to Specify How the Engine Reads and Writes Data" on page 151](#).

5. To change the sort order of the data areas, double-click any of the column headings.

4.1.2 Limiting the Data Records That Appear in the Edit Panel

A test data source might include more customers or records than you require to map the data file. If you are mapping a print file or a data file with record type indicators, you can limit your view to a range of records for a representative sample of data.

For more information about print files, see [“Setting Up Data Files to Mine Data from a Print File” on page 296](#).

To limit the data records that appear in the Edit Panel:

1. In Design Manager, from the Library, drag a print file or a data file with record type indicators to the Edit Panel.
2. Right-click the data file and select **View > Data Mapping Range**.

The **Data Mapping Range** dialog box opens.

3. To specify the first record to view, you must do the following:
 - a. Select the **Beginning record** check box.
 - b. In the adjacent box, enter the number of the first record to view.
4. To specify the last record to view, you must do the following:
 - a. Select the **Ending record** check box.
 - b. In the adjacent box, enter the number of the last record to view.
5. Click **OK**.

The **Data Mapping Range** dialog box closes and the range you specified opens in the Edit Panel.

4.1.3 Changing the Text Encoding in the Edit Panel

If you work with data that contains both SBCS and DBCS characters, invisible shift in/shift out characters surround double-byte data to identify when Design Manager must switch character sets. Design Manager recognizes additional space for these invisible characters, which can cause the data to appear out of alignment in columnar data files. Changing the encoding corrects the visual alignment of the text in the Edit Panel.

To change the text encoding in the Edit Panel:

1. In Design Manager, from the Library, drag a columnar data file to the Edit Panel.
2. On the Data Mapping toolbar, click .

The view of the data in the Edit Panel updates, but there is no effect on data mapping or content.

4.1.4 Changing the Font in the Edit Panel

You can change the font in which the data in the Edit Panel appears. You can change the font to make it easier to view or, especially if you are using DBCS characters, because the current font does not support all of the characters in the data file.

To change the font in the Edit Panel:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click the data file and select **View > Font**.
The **Font** dialog box opens.
3. Select the font, font style, and size of the font.
4. Click **OK**.

The **Font** dialog box closes and the font in the Edit Panel updates.

4.1.5 Viewing the Hexadecimal Value of the Text

Viewing the hexadecimal value of a data area helps you make sure that you are mapping the correct data. Sometimes you cannot accurately view data in a data file. For example, the Edit Panel might not display the data correctly if it the data is in a packed format or if the data uses characters that are not included in your font.

For more information about data area formats, see “[Data Area Input Formats](#)” on page 162.

To view the hexadecimal value of the text:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Highlight the text that you want to view.
3. From the shortcut menu, select **View > View HEX**.

The **HEX View** dialog box opens and shows you the hexadecimal value of each of the characters in the data area.

4.2 Automapping a Data File

Automapping lets you automatically find and map all of the data areas in a data file. When you automap a data file, Design Manager uses the data file structure in the test data source or an external definition file to find all of the data areas. Design Manager uses the sample data to define the data area input format and to associate a variable with each of the data areas. The variables used during automapping can be existing variables or they can be new variables created as part of the automapping process. If you create variables during automapping, Design Manager uses the sample data to identify the variable type and to name the variable.

For more information about the data area input format, see [“Data Area Input Formats” on page 162](#).

For more information about sample data, see [“Setting Up Data Files” on page 9](#).

The automapping process is often faster than manual mapping since it maps the data file in its entirety, but it can be less precise. For example, dates might be identified as strings, instead of the expected MM/DD/YYYY format, during automapping. Review the format settings for accuracy when automapping to ensure the output includes the data format you expect. Also, by default, when Design Manager automaps a data file, it maps all of the data areas in the data file. Before automapping, plan which data areas you must map. You are not required to map variables to all of the data areas in a data file. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory required, map only those data areas you want to use to populate variables.

For more information about manual mapping, see [“Manually Mapping a Data File” on page 146](#).

You can automap a columnar, delimited, Live, JSON, or XML formatted data file.

For more information about automatically mapping data files, see the following topics:

- [“Automapping a Columnar Data File” below](#)
- [“Automapping a Delimited Data File” on page 140](#)
- [“Automapping a JSON Formatted Data File” on page 261](#)
- [“Automapping an XML Formatted Data File” on page 190](#)

4.2.1 Automapping a Columnar Data File

You can automap a columnar data file using a copybook. Exstream Design and Production supports copybooks that use the Common Business-Oriented Language (COBOL) programming language to define the data layout. Design Manager reads the data layout in the copybook and uses the information to define the location and size of the data areas, the variable name and type, and the format of the data area. As with other types of automapping, the variables can be existing variables or the variables can be created as part of the automapping process. If you use

the copybook to create variables, keep in mind that the variables must follow the Exstream Design and Production variable naming conventions.

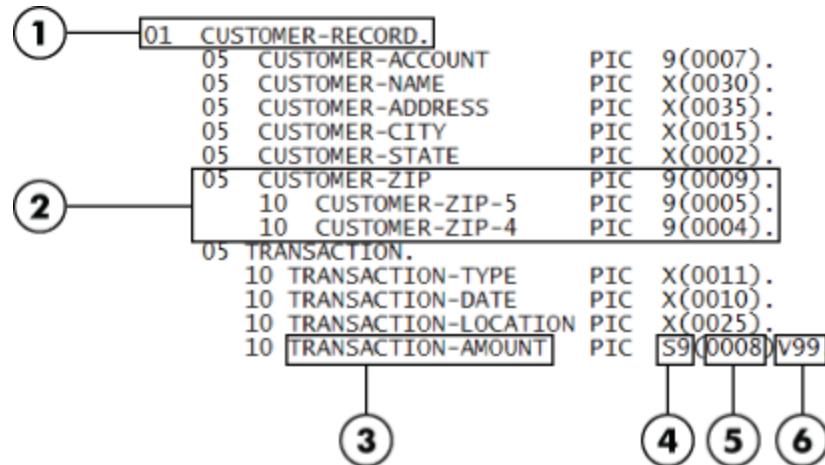
For more information about variable naming, see [“Setting Up Variables” on page 59](#).

A copybook contains the layout for a single record in a columnar data file. If your data file uses multiple record types, you must import a new copybook and specify variables for each record type you want to define. If you want to use the same variables and not import a new copybook, you can also copy the record layout from one record type to another within the same data file.

For more information about copying records, see [“Copying the Data Map Layout of a Record” on page 150](#).

Even if you do not use a COBOL compiler to programmatically generate the copybook, you can use the COBOL standards to manually create a copybook that you can then use to automap a columnar data file. To help you generate a copybook or to create one of your own, the following sample describes how Design Manager interprets the different parts of a COBOL copybook file.

Design Manager interpretation of a sample COBOL copybook



| | |
|----------|--|
| 1 | Start new record |
| 2 | Double map the data areas |
| 3 | Variable name |
| 4 | Variable type |
| 5 | Number of characters used in the data area |



Data area format

Supported COBOL Clauses

COBOL clauses define the type and format of the data in each of the data areas. Exstream Design and Production supports the following commonly used COBOL clauses:

- OCCURS—Defines a data area as a horizontal array
- PICTURE or PIC—Defines the type of data in the data area
- REDEFINES—Defines a previously defined data area, which is the equivalent of double-mapping a data area
- SIGN [LEADING] SEPARATE—Specifies that the numeric value includes a sign (positive or negative) and defines the location of the sign
- USAGE—Defines the data area format of a data area

PIC and USAGE Clauses

PIC and USAGE clauses are the most common clauses in copybooks created for Exstream Design and Production. You use them to define the type of data in the data area and the format of the data area. Design Manager uses the format characters in PIC and USAGE clauses to select the variable type to map to the data area. Design Manager also uses the PIC and USAGE clauses to select the input format of the data area.

For more information about variable types, see “[Data Type Represented by a Variable](#)” on [page 62](#).

For more information about data area formats, see “[Specifying the Input Format of Data in a Data Area](#)” on [page 154](#).

Not all of the COBOL format characters found in PIC and USAGE clauses are supported by Exstream Design and Production. The following table lists the supported format characters and their related variable types and data input formats.

Supported COBOL format characters

| Format character | Description | Variable type | Data input format |
|------------------|--|---------------|----------------------|
| B | A numeric value or a string with a blank in the specified location | String | Trim Trailing Blanks |
| CR | A numeric value that begins with a CR when it is negative and blanks when it is positive | String | Trim Trailing Blanks |

Supported COBOL format characters, continued

| Format character | Description | Variable type | Data input format |
|------------------|---|--|--|
| DB | A numeric value that begins with a DB when it is negative and blanks when it is positive | String | Trim Trailing Blanks |
| S | A numeric value with an implied sign | Integer (no decimal) Currency (decimal) | COBOL Signed (Trailing) |
| V | A numeric value with an implied decimal point | Currency | General Number |
| X | A string that is any combination of alphabetic, numeric, and symbolic characters | String | Trim Trailing Blanks |
| Z | A numeric value that suppresses leading zeros and is replaced with a blank when the value is exactly zero | String | Trim Trailing Blanks |
| 0 | A numeric value with a 0 in the specified location | String | Trim Trailing Blanks |
| 9 | A numeric value | Integer (no decimal) Currency (decimal) | General Number |
| * | A numeric value that replaces leading zeros with an * (asterisk) | String | Trim Trailing Blanks |
| , | A numeric value or a string with a , (comma) in the specified location | String | Trim Trailing Blanks |
| . | A numeric value with a . (decimal) in the specified location | Currency | General Number |
| \$ | A numeric value that begins with a \$ to the left of the most significant digit and replaces any leading zeroes | String | Trim Trailing Blanks |
| + | A numeric value that begins with a - (minus sign) when it is negative and + (plus sign) when it is positive | Integer (no decimal) Currency (decimal) | COBOL Sep-Leading Sign |
| / | A numeric value or a string with a / (slash) in the specified location | String (string) Integer (numeric) | Trim Trailing Blanks General Number |

Using a COBOL Copybook to Automap a Columnar Data File

1. In Design Manager, from the Library, drag a columnar data file to the Edit Panel.
2. In the record you want to map, right-click any data area and select **Record > Import From Copybook**.

The **Import From Cobol Copybook** dialog box opens.

3. In the **File** box, enter the name of the external copybook file you want to use.

The contents of the file appear as a tree structure in the **Select nodes to import** area.

4. To select the definitions to import, do to one of the following:

| To | Do this |
|--|--|
| Select all of the definitions in the copybook | Click Select All . |
| Select a branch of definitions in the copybook | <ol style="list-style-type: none">In the Select nodes to import box, click the parent branch.Click Select Children. |
| Select a specific definition only | In the Select nodes to import box, click the definition. |

5. Click **Next**.

The **Copybook Import Settings** dialog box opens so you can set the properties of the imported variables.

6. To add a prefix to all of the variable names that will be created during automapping, enter a string of characters in the **Variable name prefix** box. For example, you can add `Vista` to associate the variables with the Vista application.
7. In the **Folder** box, select the folder in the Library in which to store new variables or to find existing variables.

8. To define the variables to map to the data file, do the following:

| To | Do this |
|---|--|
| Map the data file using new variables | <ol style="list-style-type: none">a. Select the Map to new data items check box.b. From the Type list, select the type of data that the variable represents.c. In the Design sample record box, select the record used to define the design samples for the new variables. Exstream Design and Production does not support Unicode for design samples (optional).d. If the variable can have more than one value per customer, select the Array check box. <p>For more information about variable types, see "Types of Variables" on page 60.</p> |
| Map the data file using existing variable | Select the Map to existing data items only check box. |
| Create new variables but do not use them to map the current data file | <ol style="list-style-type: none">a. Clear the Map to new data items check box.b. Clear the Map to existing data items only check box. |

9. From the **Format** list, select an option to specify the data input format of the data area. The options vary based on the variable type mapped to the data area.
10. Click **Import**.

Design Manager automaps the data file.

After you automap a data file, you can make adjustments to the variable and data area properties and to the data map of specific data areas.

For more information about variable properties, see "[Setting Up Variables](#)" on page 59.

For more information about data area properties, see "[Using a Data Area to Specify How the Engine Reads and Writes Data](#)" on page 151.

For more information about adjusting individual data areas, see "[Modifying a Data Map](#)" on page 157.

4.2.2 Automapping a Delimited Data File

You can extract data from a database into a delimited data file. During extraction, you can add a header definition that describes the data to the top of the data file or store the header definition in an external header definition file. You can use the header definition to automap the data file.

When you automap a delimited data file, Design Manager uses the header definition to automatically find areas in the data file.

If you create variables as part of the data mapping process, Design Manager uses the header definition to name any variables it creates. Design Manager also uses the first record to define the variable type and to select a design sample. For best results, use the following table to determine the format of the data you want to include in the first record.

Variable type definition

| If you want this variable type | Include this format of data in the record |
|--------------------------------|--|
| Boolean | One of the following strings: True, False, Yes, No |
| Currency | A number with two digits after a decimal |
| Date | A date in one of the following formats: MM/DD/YYYY, DD/MM/YYYY, YYYY/MM/DD, YYYY/DD/MM, MM-DD-YYYY, DD-MM-YYYY, YYYY-MM-DD, YYYY-DD-MM |
| Floating | A number with any number of digits other than two and less than six |
| Integer | A number with no decimals |
| String | Any string of characters. The variable type defaults to string if no other variable type matches. |

To automap a delimited data file, you must complete the following tasks:

1. [“Setting Up a Delimited Data File Object for Automapping” below](#)
2. [“Automapping a Delimited Data File” on the previous page](#)

Setting Up a Delimited Data File Object for Automapping

Before you automap a delimited data file, compare the data file and the header definition to make sure that they have the same number of delimited data areas and use the same delimiter. In addition, make sure that the data file does not use horizontal arrays or repeating groups because Design Manager cannot automap data files that repeat data in a single record.

For more information about array variables, see [“Number of Values a Variable Can Represent” on page 63](#).

For more information about repeating groups, see [“Mapping Groups of Data Areas That Repeat in a Single Record” on page 149](#).

Lastly, before you automap a delimited data file, you must also make sure that you have set the appropriate data file properties. If the appropriate data file properties are not set, Design Manager cannot use your header definition to automap the data file.

To set up the delimited data file object for automapping, complete one of the following tasks:

- [“Setting Up Customer Driver Files and Reference Files” on the next page](#)
- [“Setting Up Initialization Files and Post-sort Initialization Files” on the next page](#)
- [“Setting Up Report Files and Post-Sort Report Files” on page 143](#)
- [“Setting Up Auxiliary Layout Files” on page 143](#)

Setting Up Customer Driver Files and Reference Files

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. Verify that the header definition is in the correct location based on the option you selected from the **How to identify customers in the data file** list.

| If you selected this option | Specify the header definition in this location |
|---|--|
| Each customer has the same number of records | <ul style="list-style-type: none">Within the data file only if there is one record per customerIn an external header definition file for any number of records per customer |
| A new customer occurs on specified record type(s) | <ul style="list-style-type: none">In an external header definition file |
| A customer ID is on every record | <ul style="list-style-type: none">Within the data fileIn an external header definition file |

4. Go to the **Advanced** tab.
5. In the **Header** box, enter 0 to specify that you do not want to ignore headers. If you enter a value other than 0, automapping is disabled.
6. If the data file you are automapping has the header definition within the data file, verify that the production data file does not include the header definition. Production data files treat header definitions as data records.
7. From the **Edit** menu, select **Save**.

Setting Up Initialization Files and Post-sort Initialization Files

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. Verify that the header definition is in the correct location based on the option you selected from the **How data is identified in the data file** list.

| If you selected this option | Specify the header definition in this location |
|--|--|
| Each record in the file has separate data mappings | <ul style="list-style-type: none">In an external header definition file if the number of header records matches the number of records in the data file |
| Each has a specified record type(s) | <ul style="list-style-type: none">In an external header definition file |

| If you selected this option | Specify the header definition in this location |
|----------------------------------|--|
| Each record has the same mapping | <ul style="list-style-type: none">• Within the data file• In an external header definition file |

4. Go to the **Advanced** tab.
5. In the **Header** box, enter 0 to specify that you do not want to ignore headers. If you enter a value other than 0, automapping is disabled.
6. If the data file you are automapping has the header definition within the data file, verify that the production data file does not include the header definition. Production data files treat header definitions as data records.
7. From the **Edit** menu, select **Save**.

Setting Up Report Files and Post-Sort Report Files

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. Verify that the header definition is specified within the data.
4. Go to the **Advanced** tab.
5. In the **Header** box, enter 0 to specify that you do not want to ignore headers. If you enter a value other than 0, automapping is disabled.
6. If the data file you are automapping has the header definition within the data file, verify that the production data file does not include the header definition. Production data files treat header definitions as data records.
7. From the **Edit** menu, select **Save**.

Setting Up Auxiliary Layout Files

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Advanced** tab.
3. In the **Header** box, enter 0 to specify that you do not want to ignore headers. If you enter a value other than 0, automapping is disabled.
4. If the data file you are automapping has the header definition within the data file, verify that the production data file does not include the header definition. Production data files treat header definitions as data records.
5. From the **Edit** menu, select **Save**.

Automapping a Delimited Data File

1. In Design Manager, from the Library, drag the data file to the Edit Panel.
2. Right-click the Edit Panel and select **Auto Map using header record(s)**.

The **Automap a delimited data file** dialog box opens.

3. To select the location of the header definition, do one of the following:

| To | Do this |
|---|---|
| Use a header definition stored in the data file | Select the Use header definition from data file radio button. |
| Use an external header definition file | <ol style="list-style-type: none">a. Select the Use header definition from header definition file radio button.b. In the Header File box, select the location of the header definition file. |

4. If you have one or more data areas with existing maps, select one of the following radio buttons:
 - **Retain existing maps and create new for the rest**—Preserve the existing data maps and automap only the unmapped data areas.
 - **Delete existing maps and create new for all**—Remove any existing data maps and automap all data areas, including those with existing data maps.
5. To select the variables to use when automapping, you must do one of the following:

| To | Do this |
|---------------------------------------|--|
| Map the data file using new variables | <ol style="list-style-type: none">a. In the Variable mapping options area, select the Map to new variables radio button.b. In the Mapped variables folder box, select the folder in which to store the new variables.c. Click Next. The Automap a delimited data file dialog box updates. Data areas in the header definition with errors are highlighted in red in the Header definitions box.d. To add a prefix to all of the variable names (for example, to associate the variables with the particular data file) that will be created during automapping, you must do the following:<ol style="list-style-type: none">i. In the Variable name prefix box, enter a string of characters.ii. Click Apply Prefix. |

| To | Do this |
|--|---|
| Map the data file using existing variables | <ol style="list-style-type: none">In the Variable mapping options area, select the Map to existing variables radio button.In the Mapped variables folder box, select the folder in which the existing variables are stored.Click Next. <p>The Automap a delimited data file dialog box updates. Data areas in the header definition with errors are highlighted in red in the Header definitions box.</p> |

6. To ignore data areas when automapping, do one or both of the following:

| To | Do this |
|---------------------------------------|--|
| Ignore a specific data area | <ol style="list-style-type: none">In the Header definitions box, click the data area you want to ignore.Click the Ignore check box. |
| Ignore header definitions with errors | Click Ignore all errors . |

7. In the **Variable settings** area, verify and, if necessary, update variable properties. If the variable can have more than one value per customer, select the **Array** check box. Keep in mind that horizontal arrays are not supported.
8. In the **Data area settings** area, verify and, if necessary, update the data area format properties.
9. Click **Finish**.

The **Automap a delimited data file** dialog box closes and Design Manager automaps the data file.

After automapping a data file, you can make adjustments to the variable and data area properties and to the data map of individual data areas.

For more information about variable properties, see “[Setting Up Variables](#)” on page 59.

For more information about data area properties, see “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151.

For more information about adjusting individual data areas, see “[Modifying a Data Map](#)” on page 157.

4.3 Manually Mapping a Data File

Manually mapping a data file involves a series of steps in which you associate a variable with individual data areas and groups of data areas. You are not required to map variables to all of the data areas in a data file. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory required, map only those data areas you want to use to populate variables.

To manually map a data file, complete the following tasks as needed:

- “[Mapping a Variable to Sample Data in a Data Area](#)” below
- “[Mapping a Data Area with No Sample Data in a Delimited Data File](#)” on the next page
- “[Mapping Multiple Variables to a Single Data Area](#)” on page 148
- “[Mapping Groups of Data Areas That Repeat in a Single Record](#)” on page 149
- “[Copying the Data Map Layout of a Record](#)” on page 150

4.3.1 Mapping a Variable to Sample Data in a Data Area

Before the engine can use data from a data area, you must map the data area to a variable. If you are mapping a data area that corresponds to an existing variable, you can directly map the variable to the data area. If no variable exists, you must create a variable. Rather than exiting the data mapping process, you can select the data area first and then create the variable.

To map a variable to sample data in a data area:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Highlight an unmapped (gray) data area that you want to map. For columnar data files, click and drag to highlight. For all other data file formats, double-click to highlight.
3. To map a variable to the data area, you must do one of the following:

| To | Do this |
|--------------------------|---|
| Map an existing variable | <ol style="list-style-type: none">a. Right-click the area and select Data Area > Map Existing Data Item. The Select Variable dialog box opens.b. Double-click the appropriate variable. The Data Area Properties dialog box opens. |

| To | Do this |
|--------------------|--|
| Map a new variable | <ol style="list-style-type: none">a. Right-click the area and select Data Area > Map New Data Item. The New Variable dialog box opens.b. In the Name box, enter a name. In the Description box, enter a description (optional).c. From the Type list, select the type of data the variable represents.d. If the variable can contain more than one value, select the Array check box.e. In the Design sample box, enter the text string you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional).f. Click Finish. <p>The New Variable dialog box closes and the Data Area Properties dialog box opens.</p> <p>For more information about variable properties, see “Setting Up Variables” on page 59.</p> |

4. In the **Data Area Properties** dialog box, specify the input properties of the data area being mapped.

For more information about the input properties, see “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151.

5. Click **OK**.

The **Data Area Properties** dialog box closes and the highlights the data area, based on its use, to indicate that it has been mapped to a variable. In addition, you can view information (such as the name) about the variable when you place your pointer over the data area. If you want information about what the highlighting colors mean in the Edit Panel, right-click the Edit Panel and select **View > What do colors mean**.

4.3.2 Mapping a Data Area with No Sample Data in a Delimited Data File

In delimited data files, when a customer does not have data, consecutive delimiting characters appear with no data between them, which creates an empty data area. For example, suppose you are mapping a record that includes a customer address. If the customer does not have a second address line, the data area is empty.

Even if the data area is empty in the sample data, you can map and identify the type of data that should appear between the two delimiters. As with other data areas, you must map the empty data area if you want to include the data during production. If you are mapping an empty data area, you must map an existing variable. You cannot create a new variable as part of the data mapping process.

To map an empty data area in a delimited data file:

1. In Design Manager, from the Library, drag a delimited data file to the Edit Panel.
2. Highlight anywhere in the record you are mapping.
3. Right-click and select **Data Area > Add Delimited Item**.

The **Select Variable** dialog box opens.

4. Double-click the desired variable.

The **Delimited Item to Specify** dialog box opens.

5. In the **Item number** box, enter the sequence number of the empty data area. As you count left to right, begin at zero in the first data area.
6. Click **OK**.

The **Data Area Properties** dialog box opens.

7. In the **Data Area Properties** dialog box, specify the input properties of the data area being mapped.

For more information about the input properties, see “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151.

8. Click **OK**.

The **Data Area Properties** dialog box closes.

4.3.3 Mapping Multiple Variables to a Single Data Area

Data areas can have more than one purpose, each of which might require more than one variable. For example, you can use the data in a data area directly on a page to customize a design; at the same time, you can use the data in a formula to select an insert in a mailing. Rather than having the same information appear in the data file multiple times, you can map multiple variables to the same data area.

To map multiple variables to a data area:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Map a variable to the data area.

The data area is highlighted according to its use.

3. Highlight the data area.
4. Map a second variable to the data area.

The data area color changes to cyan to show that it is mapped to more than one variable. In addition, the information about the data area updates to specify that there is an overlap and lists all of the variables mapped to the data area.

For more information about mapping a variable to a data file, see [“Mapping a Variable to Sample Data in a Data Area” on page 146](#).

4.3.4 Mapping Groups of Data Areas That Repeat in a Single Record

A set of data areas that repeats in a single record is called a repeating group. Repeating groups are often used with sets of transactions that repeat as a group in a transaction table for each customer. Because of how the engine processes repeating groups, creating a repeating group might require less memory than individually mapped data areas. The number of repeating groups in a record can be constant or the number can differ from customer to customer based on the value of a variable. For example, your data might list all of the transactions on a single date in a single record line.

You can nest groups that have a constant number of data areas within groups that have a variable number of data areas. You can also nest groups that have a variable number of data areas within groups that have a constant number of data areas.

To map groups of data areas that repeat in a single record:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a mapped data area and select **Data Area > Repeating Group**.

The **Repeating group** dialog box opens.

3. From the **Repeating group** list, you must do one of the following:
 - To create a new repeating group, select **new**.
 - To update an existing repeating group, select one of the available options. The options are named using the names of the variables mapped to the first data area and the last data area in the repeating group.
4. From the **Start data area** list, select the variable name mapped to the first data area in the group.
5. From the **End data area** list, select the variable name mapped to the last data area in the group.

6. To specify the repeat method, you must do one of the following:

| To | Do this |
|---|---|
| Specify that the number of repeating groups is constant | <ol style="list-style-type: none">From the Method list, select Constant number.In the Constant count box, enter the number of repeating groups. The number you enter must be greater than 1. |
| Specify that the number of repeating groups is constant, but the engine stops reading data when it reaches an empty value for the start data area | <ol style="list-style-type: none">From the Method list, select Constant-until empty.In the Constant count box, enter the number of repeating groups. <p>Note: A value is empty if it is a null value, spaces only, or (for Boolean, currency, and integer) a zero.</p> |
| Specify that the number of repeating groups is constant, but the engine selects groups based on a variable | <ol style="list-style-type: none">From the Method list, select Constant-use variable number.In the Constant count box, enter the number of repeating groups.In the Variable box, enter the controlling variable. |
| Specify that the number of repeating group varies based on a variable | <ol style="list-style-type: none">From the Method list, select Variable number of sets.In the Variable box, enter the controlling variable. |
| Reset the highlighted area so it is not a repeating group | From the Method list, select Ignore . |

7. Click **OK**.

The **Repeating Groups** dialog box closes and a red line appears around the data areas in the repeating group.

4.3.5 Copying the Data Map Layout of a Record

If you are mapping a data file with multiple types of record indicators, you can copy the layout of a data map in one record and use the copied layout in another record to make mapping faster.

For example, suppose you are mapping a data file with a section for each type of transaction, but you want to use the same variables for each section. You can copy the layout of one section then use it to map another section.

To copy the data map layout of a record:

- In Design Manager, from the Library, drag a data file to the Edit Panel.
- Map the first record.
- Highlight anywhere in the second record.

4. Right-click and select **Record > Copy record**.

The **Select the Record Layout to Copy** dialog box opens.

5. Select the record layout you want to copy.
6. Click **OK**.

The **Select the Record Layout to Copy** dialog box closes and the record layout from the first record is copied into the second record.

For more information about mapping a variable to a data file, see “[Mapping a Variable to Sample Data in a Data Area](#)” on page 146.

4.4 Using a Data Area to Specify How the Engine Reads and Writes Data

A data area is a single piece of data within the data file. While the properties of a variable define data during output, the properties of a data area tell the engine how and when to read data. Data areas also define how to separate different types of data from each other. If you have automapped a data file, you can adjust the default data area properties. When you define a data area, complete the following tasks as needed:

- “[Starting a New Customer, a New Section, or a New Recipient](#)” below
- “[Specifying How the Engine Uses a Data Area During Processing](#)” on page 153
- “[Specifying the Input Format of Data in a Data Area](#)” on page 154
- “[Specifying When the Engine Writes Data to a Report File](#)” on page 156

For more information about variable properties, see “[Setting Up Variables](#)” on page 59.

4.4.1 Starting a New Customer, a New Section, or a New Recipient

Note: This task assumes that you are mapping data files of any format, with the exception of XML formatted data files. For more information about mapping XML formatted data files, see “[Mapping an XML Formatted Data File](#)” on page 189.

Unless every customer has the same number of records, such as in a customer driver file with only customer contact information, you must identify where a new customer starts. If you have a data file with section data, you must also identify where a new section or subsection starts. Also, when you are using recipient data, you must identify where a new recipient starts.

Whether it is a customer, a section, or a recipient, you must specify the start using record type indicators, which are highlighted in red when you open a data file in the Edit Panel.

For more information about sections, see “[Section Data in Exstream Design and Production](#)” on [page 4](#).

For more information about adding recipient copies of documents, see *Designing Customer Communications* in the Exstream Design and Production documentation.

For more information about record type indicators, see “[Parts of a Data File](#)” on [page 16](#).

To specify the start of a new customer or a new section:

1. In Design Manager, from the Library, drag a data file to the Property Panel and verify that the data file uses record type indicators.
2. From the Library, drag the data file to the Edit Panel.

The record type indicators are highlighted in red.

3. Highlight the record indicator you want to define.
4. Right-click and select **Record > Record Properties**.

The **Record Properties** dialog box opens.

5. To force the engine to skip the indicator, select the **Ignore** check box. For example, you ignore a record when none of the data in the record is used during production.
6. Based on the type of indicator you are defining, do one or more of the following:

| To | Do this |
|-------------------------------------|---|
| Define the start of a new customer | Select the Start new customer check box. |
| Define the start of a new section | <ol style="list-style-type: none">a. Select the Start new section check box.b. In the adjacent box, enter the name of the section.c. If you want the identifier to start a new section only the first time it appears in the customer, select the First occurrence only check box. |
| Define the start of a new recipient | Select the Start new recipient check box. |
| Define a sub-indicator | <ol style="list-style-type: none">a. In the Sub-indicator area, select the Use sub-indicator check box.b. In the Start box, enter the starting column (for columnar files) or the starting field (for delimited files).c. In the Length box, enter the length of the indicator. |

7. Click **OK**.

The **Record Properties** dialog box closes. If you start a new customer, a solid black line appears above the indicator. If you start a new section or a new recipient, a dashed black and red line appears above the indicator.

4.4.2 Specifying How the Engine Uses a Data Area During Processing

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a mapped data area and select **Data Area > Data Area Properties**.

The **Data Area Properties** dialog box opens.

3. If you are mapping a customer driver file that can be updated during processing or a sort index file, you must do one of the following to specify the use of the data area:

| To | Do this |
|---|--|
| Specify that the data area is read from during processing | From the Use list, select Input . |
| Specify that the data area is written to during processing | <ol style="list-style-type: none">a. From the Use list, select Output.b. In the Output justification area, select one of the following radio buttons to specify the alignment of the data:<ul style="list-style-type: none">• None—Align the data as it appears.• Left—Align the data to the left.• Right—Align the data to the right.c. In the Output padding area, select one of the following radio buttons, to specify the method to fill unused space in the data area:<ul style="list-style-type: none">• Space—Pad the data area with spaces.• Zero—Pad the data area with zeros. |
| Specify that the data area is both read from and written to during processing | <ol style="list-style-type: none">a. From the Use list, select Update.b. In the Output justification area, select one of the following radio buttons to specify the alignment of the data:<ul style="list-style-type: none">• None—Align the data as it appears.• Left—Align the data to the left.• Right—Align the data to the right.c. In the Output padding area, select one of the following radio buttons, to specify the method to fill unused space in the data area:<ul style="list-style-type: none">• Space—Pad the data area with spaces.• Zero—Pad the data area with zeros. |
| Specify that the data area sets the initial value of the variable | From the Use list, select Initialize . |

For more information about setting up customer driver files that can be updated during processing, see “[Selecting the Customer Driver File Data Source](#)” on page 27.

For more information about sort index files, see *Creating Output* in the Exstream Design and Production documentation.

4. From the **Action if data not found** list, select one of the following options to specify what the engine does if the data is not found in the area during processing:
 - **Ignore**—Ignore missing data and continue processing.
 - **Message**—Issue a message and continue processing.
 - **Message and stop**—Issue a message and stop processing.
 - **Message, skip document**—Issue a message and skip the current customer.

5. If the data area contains confidential information, select **Design sample** from the **Make confidential using** list.

The data area is marked as confidential but the output does not change at run time unless you use the SECUREOUTPUT engine switch. To produce a new data file that does not contain confidential data, use the COPYINPUT engine switch.

For more information about the SECUREOUTPUT and COPYINPUT engine switches, see *Switch Reference* in the Exstream Design and Production documentation.

6. Click **OK**.

The **Data Area Properties** dialog box closes.

4.4.3 Specifying the Input Format of Data in a Data Area

When you map a data file, you must specify the format of the data area to which you are mapping the variable. The data input format tells the engine how to read the data. Data areas can represent data that requires the engine to adjust how it reads the data. For example, if you are mapping a Boolean variable, you must specify the value that represents true and the value that represents false.

To specify the input format of data in a data area, perform the following steps:

1. In Design Manager, from the Library, drag a data file to the **Edit Panel**.
2. Right-click a mapped data area and select **Data Area > Data Area Properties**.

The **Data Area Properties** dialog box opens.

3. From the **Format** list, select an option to specify the format of the data area. The options vary based on the variable type mapped to the data area.

For more information about the available format options, see “[Data Area Input Formats](#)” on [page 162](#).

4. If you selected **Placeholder Content** or **Composed XML (DXF) Content** from the **Format** list, select one of the following options from the **Encoding** list:
 - **None**—The content imported by the variable is not encoded.
 - **Base-64**—The content imported by the variable uses Base-64 encoding.
5. In the **Start** box, enter the starting point of the highlighted area in the data file. If you are mapping a delimited data file, the start is the sequence number, starting at one, of the element being mapped.
6. In the **Length** box, enter the width of the highlighted area in the data file. If you are mapping a delimited data file, the length is 0.
7. If you are mapping a currency or a floating variable and you want to specify the location of the decimal when it does not appear in the data, enter the number of implied digits in the **Implied digits** box. Enter a positive number (for example, 1 or 5) to move the decimal to the left. Enter a negative number (for example, -1 or -5) to move the decimal to the right. You can have a maximum of nine implied digits.
8. If you are mapping a date, you can change the century cutoff year by selecting the **Century cutoff** check box and entering a two-digit year (00-99) that indicates the year you want to use for the century cutoff.

Note: The century cutoff year is used to interpret how years in two-digit format (YY) should be transposed into four-digit format (YYYY) and which century they belong to. Two-digit years less than the specified value are considered to fall in the current century (20xx); two-digit years greater than or equal to this value fall in the previous century (19xx). The default value is 20.

The century cutoff can also be changed using the TWO_DIGIT_CENTURY_LIMIT switch. For more information, see *Switch Reference* in the Exstream Design and Production documentation.

9. If you are mapping an output data file (a report file, for example) and if the variable being mapped is a static array, you must do one of the following:

| To | Do this |
|--|---|
| Indicate a specific array element when an array is defined horizontally on a row or is in a non-sequential order | <ol style="list-style-type: none">a. Select the Specific array element check box.b. In the adjacent box, enter the element number of the element. |
| Use the next available array element | Clear the Specific array element check box. |

10. If you are mapping a data area that contains DBCS characters and the width of characters can vary in the input but you want them consistent in the output, do the following:
 - a. Click **Conversions**.
The **Conversion Properties** dialog box opens.
 - b. Select the type of conversion for numbers, alpha, and katakana characters.
 - c. If your data files uses a backslash to represent the yen sign, select the **Convert Half-Width Backslash to Full-Width Yen Sign** check box.
 - d. Click **OK**.
The **Conversion Properties** dialog box closes.
11. If you are mapping a columnar customer driver, initialization, or reference data file with a mix of SBCS and DBCS characters and the data file does not include shift out/shift in characters to identify a change from SBCS to DBCS fields, select the **Add Shift Out and Shift In** check box to add indicators to the beginning and end of the data area.
12. Click **OK**.
The **Data Area Properties** dialog box closes.

4.4.4 Specifying When the Engine Writes Data to a Report File

You can change the engine timing so that the engine writes data to a report file or a post-sort report file at different times. You can define the engine timing either in the data file properties or in the record properties. You define the engine timing in the data file when you want to apply the same timing to all of the records in the data file. You define the engine timing on individual records when you want to apply specific timing to each of the records.

For more information about defining the engine timing in a report or post-sort report file, see [“Defining the Records in a Report or Post-sort Report File” on page 55](#).

To specify when the engine writes data to a report file:

1. In Design Manager, from the Library, drag a report file to the Property Panel.
2. Go to the **Advanced** tab.
3. From the **IO time** list, select **As determined by record properties**.
4. From the **File** menu, select **Save**.
5. From the Library, drag the report file to the Edit Panel.
6. Highlight the record you want to define.
7. Right-click and select **Record > Record Properties**.

The **Record Properties** dialog box opens.

8. From the **IO time** list, select an option to specify when the engine reads data records.
9. Click **OK**.

The **Record Properties** dialog box closes.

4.5 Modifying a Data Map

After you automap or manually map a data file, you can still make changes to the data map. For example, the sample data had an area that was too short for the actual data or you made a mistake and mapped a variable to the wrong data area.

To modify a data map, complete the following tasks as needed:

- “[Resizing the Length of a Data Map in a Columnar Data File](#)” below
- “[Moving a Data Map](#)” on the next page
- “[Mapping a New Variable to a Data Area](#)” on the next page
- “[Removing a Data Map](#)” on page 159

4.5.1 Resizing the Length of a Data Map in a Columnar Data File

If you are mapping a columnar data file and find that a data map is too long or too short, you can resize the data map to accurately reflect the actual length of the data.

To resize the length a data map in a columnar data file:

1. In Design Manager, from the Library, drag a columnar data file to the Edit Panel.
2. Click a mapped area to make it active.
A solid line appears around the data area.
3. Click and drag in the data area to highlight the new length of the data area.
4. Right-click and select **Data Area > Reset Data Area Location**.

The data area resizes to the highlighted length.

4.5.2 Moving a Data Map

If you mapped a variable to the wrong data area, you can move the data map to a different location. You can also move data areas when the sample data area is updated to include new data areas or to remove unused data areas.

To move a data area:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Click a mapped area to make it active.

A solid line appears around the data area.

3. Right-click the area and select **Record > Move data areas**.

The **Move Data Areas** dialog box opens.

4. In the **Move** box, enter the number of columns or fields you want to move.
5. To move the data area to the left, select the **Move left** check box.
6. Click **OK**.

The **Move Data Areas** dialog box closes and the data area moves to the new location.

Note: If you move a data map in a columnar data file or a delimited data file, all of the data areas that adjoin the selected data area in the direction of the move, will also be moved. For example, if you move a mapped area to a new location to the left of its original location, all of the data areas to the left of the original location will also be moved along with the data area that you want move.

4.5.3 Mapping a New Variable to a Data Area

If you have mapped the wrong variable to a data area, you can select a new variable without remapping the area.

To map a new variable to a data area:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Click a mapped area to make it active.

A solid line appears around the data area.

3. Right-click and select **Data Area >Data Area Properties**.
4. In the **Variable** box, select a new variable.
5. Click **OK**.

The **Data Area Properties** dialog box closes and the new variable is mapped to the data area.

4.5.4 Removing a Data Map

If the data map for an area is incorrect, or no longer needed, you can completely remove the data map. Removing the map does not change the data area itself.

To remove a data map:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a data map and select **Data Area > Delete Data Item**.

The data area is unmapped and highlighting is removed.

4.6 Testing a Data Map

You can test a selected variable or all the variables in a data file to make sure you mapped variables correctly. Rather than completing a test run of the engine, you can verify the data map of a data file directly from the Edit Panel. The test shows the corresponding information in the data file for one or more customers.

To test a data map:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Click a mapped area to make it active.

A solid line appears around the data area.

3. Right-click and select **Test**.

The **Variable Test Results** dialog box opens.

4. To specify the values you want to return, do one of following:
 - To return the value of only the highlighted data area, select the **Selected only** check box.
 - To return the value of all the variables in the data file, clear the **Selected only** check box.
5. To return the location where each customer ends, select the **Show breaks** check box.
6. To limit the number of customers used in the test, you must do the following:

- a. Select the **Limit sets** check box.
 - b. In the boxes, enter the first and last customer to test.
7. Click **Go**.
- The test results appear in the **Results** box.
8. To view a specific variable's mapping, you must do the following:
 - a. Click a line in the **Results** box.
 - b. Click **Find**.
- The variable is highlighted in the Edit Panel.

4.7 Searching a Mapped Data File

As you move the pointer through a data file in the Edit Panel, record and column numbers appear in the bottom left corner of the Design Manager window. This information lets you quickly determine the location of data areas and record type indicators. Sometimes, however, that information is not enough: you need something more specific.

To search a mapped data file, complete the following tasks as needed:

- [“Finding Specific Text” below](#)
- [“Finding a Specific Variable” on the next page](#)
- [“Finding the Start of a Customer” on page 162](#)
- [“Finding the Start of a Section” on page 162](#)

4.7.1 Finding Specific Text

You can find specific text in a data file. For example, suppose that you know the sample text to which you must map a variable, but you cannot find the data area with the sample text in the Edit Panel. You can search for the sample text.

To find specific text:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
 2. Right-click and select **Find > Text**.
- The **Find** dialog box opens.
3. In the **Find** box, enter the text for which you want to search.

4. To search the file, do one of the following:
 - To search the entire file, select the **Wrap** check box.
 - To search from the current location to the end of the file, clear the **Wrap** check box.
 5. To find only the instances that exactly match the uppercase and lowercase characteristics of the text in the **Find** box, select the **Match case** check box.
 6. If you entered a hexadecimal value in the **Find** box, select the **HEX** check box to search for the hexadecimal value.
 7. Click **OK**.
- The first occurrence of the text is highlighted in the Edit Panel.
8. To find the next occurrence of the text, right-click in the Edit Panel and select **Find > Next Text**. The next occurrence of the text is highlighted in the Edit Panel.

4.7.2 Finding a Specific Variable

You can find a particular variable within a mapped data file. For example, suppose that you want to update the variable properties. You can search the data file for the variable to confirm that the update will not conflict with the data area to which the variable is mapped.

To find a specific variable:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. To find a variable, you must do one of the following:

| To | Do this |
|---|--|
| Find each occurrence of a variable individually | <ol style="list-style-type: none">a. Right-click and select Find > Variable. The Select Variable dialog box opens.b. Double-click the variable you want to find.c. Click OK. The first occurrence of the variable is highlighted in the Edit Panel.d. To find the next occurrence of the variable, right-click in the Edit Panel and select Find > Next Variable. The next occurrence of the variable is highlighted in the Edit Panel. |
| Find all occurrences of a selected variable | <ol style="list-style-type: none">a. Click a mapped area to make it active. A solid line appears around the data area.b. Right-click and select Find > Selected Variable. A solid line appears around all of the data areas that are mapped to the same variable. |

4.7.3 Finding the Start of a Customer

If you have specified how to start new customers in the data file, you can find where another customer starts. For example, if customer data can have multiple types of sections, you can find another customer so you can map all of sections possible in the data file.

To find the start of a customer:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. To find the beginning of the previous customer, right-click and select **Find > Prev Customer**.
3. To find the beginning of the next customer, right-click and select **Find > Next Customer**.

4.7.4 Finding the Start of a Section

If you have section data and have specified the start of a new section, you can find where the next section starts. For example, if the section you are mapping has more records than you can view in the Edit Panel, you can search for the start of another section.

To find the start of a section:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. To find the beginning of the previous section, right-click and select **Find > Prev Section**.
3. To find the beginning of the next section, right-click and select **Find > Next Section**.

4.8 Data Area Input Formats

When you map a data file, you must specify the format of the data area to which you are mapping the variable. Some data areas might require the engine to adjust how it reads the data. For example, suppose that you are mapping a string that includes extra spaces at the beginning and end. In order to process the data correctly, you can specify that you want the engine to ignore the spaces when reading the data. To specify how the engine should read the data in the data area, select the format from the **Format** list on the **Data Area Properties** dialog box.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

The format in which the engine reads the data is not required to match the format of the data in the output. You select the data output format on the variable.

For information about selecting the data output format, see “[Formatting Variable Values in Output](#)” on page 81.

You select the format of the data area using the data type associated with the variable you have mapped to the data area. To help you select the correct format, this chapter discusses the following topics:

- “[Boolean Data Area Input Formats](#)” below
- “[Currency, Floating, and Integer Data Area Input Formats](#)” on the next page
- “[Date Data Area Input Formats](#)” on page 167
- “[Formatted Text Data Area Input Formats](#)” on page 167
- “[Placeholder Data Area Input Formats](#)” on page 168
- “[String and Tagged Text Data Area Input Formats](#)” on page 172
- “[Creating Custom Data Area Input Formats](#)” on page 176

4.8.1 Boolean Data Area Input Formats

The following table lists the formats available when you are mapping a Boolean variable to a data area used for programming logic.

For more information about variable types, see “[Types of Variables](#)” on page 60.

Boolean formats

| Format | Data area description |
|------------------------|---|
| T/F | When a condition is true, T appears in the data file. When a condition is false, F appears in the data file. |
| Y/N | When a condition is true, Y appears in the data file. When a condition is false, N appears in the data file. |
| I/O | When a condition is true, 1 appears in the data file. When a condition is false, 0 appears in the data file. |
| True/False | When a condition is true, True appears in the data file. When a condition is false, False appears in the data file. |
| Yes/No | When a condition is true, Yes appears in the data file. When a condition is false, No appears in the data file. |
| General Number | The data area is Arabic numeral data. |
| Packed (Left to Right) | The engine reads the data area from the left half-byte of the first byte to the right half-byte of the last byte. |
| Packed (Right to Left) | The engine reads the data area from the right half-byte of the first byte to the left half-byte of the last byte. |

Boolean formats, continued

| Format | Data area description |
|------------------------|--|
| Packed (Left to Left) | The engine reads the data area from the left half-byte of the first byte to the left half-byte of the last byte. |
| Packed (Right to Left) | The engine reads the data area from the right half-byte of the first byte to the left half-byte of the last byte. |
| Zoned | The data area is a decimal or a currency formatted field on a mainframe system. |
| Binary Byte | The data area is a numeric value that is stored in a single byte. The first bit is treated as a plus or minus sign. The value range is -127 to +127. |
| Binary Unsigned Byte | The data area is a numeric value that is stored in a single byte. The first bit does not contain a sign character. |
| Binary Short | The data area is a numeric value that is stored in two bytes. |
| Binary Unsigned Short | The data area is a numeric value that is stored in two bytes and does not contain any sign characters. |
| Binary Integer | The data area is an integer representation that is stored in binary format. |
| Binary 64-bit Integer | The data area is a 64-bit integer representation that is stored in binary format. Apply this format to any COBOL COMP numbers with more than nine digits. Note: If you have more than four implied decimals when using this option, the engine issues a warning. |

4.8.2 Currency, Floating, and Integer Data Area Input Formats

The following table lists the formats available when you are mapping a currency, floating, or an integer variable to a data area used for numeric values.

For more information about variable types, see “[Types of Variables](#)” on page 60.

Currency, floating, and integer formats

| Format | Data area description | Data type |
|--------|--|---|
| Custom | The data area uses a customized formatting. For more information about creating a custom format, see “ Creating Custom Data Area Input Formats ” on page 176. | <ul style="list-style-type: none">• Currency• Floating• Integer |

Currency, floating, and integer formats, continued

| Format | Data area description | Data type |
|-------------------------------|--|---|
| General Number | The data area is Arabic numeral data. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Packed (Left to Right) | The engine reads the data area from the left half-byte of the first byte to the right half-byte of the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Packed (Right to Right) | The engine reads the data area from the right half-byte of the first byte to the right half-byte of the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Packed (Left to Left) | The engine reads the data area from the left half-byte of the first byte to the left half-byte of the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Packed (Right to Left) | The engine reads the data area from the right half-byte of the first byte to the left half-byte of the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Zoned | The data area is a decimal or a currency formatted field on a mainframe system. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL Signed (Trailing) | The data area is EBCDIC formatted COBOL data that has a plus or minus sign in the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL Signed (Trailing ASCII) | The data area is ASCII formatted COBOL data that has a plus or minus sign in the last byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL Signed (Leading) | The data area is COBOL data that has a plus or minus sign in the first byte. | <ul style="list-style-type: none">• Currency• Floating• Integer |

Currency, floating, and integer formats, continued

| Format | Data area description | Data type |
|------------------------------|--|---|
| COBOL Sep-Leading Sign | The data area is COBOL data that has a separate plus or minus sign at the beginning. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL Separate Sign | The data area is COBOL data that has a separate plus or minus sign at the end. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL PHASE3 Signed | The data area is COBOL data in a proprietary format that has a separate plus or minus sign at the beginning. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| COBOL PHASE3 Signed Trailing | The data area is COBOL data in a proprietary format that has a separate plus or minus sign at the end. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Hexadecimal String | The data area is data written in hexadecimal format. | <ul style="list-style-type: none">• Integer |
| Binary Byte | The data area is a numeric value that is stored in a single byte. The first bit is treated as a plus or minus sign. The value range is -127 to +127. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Binary Unsigned Byte | The data area is a numeric value that is stored in a single byte. The first bit does not contain a sign character. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Binary Short | The data area is a numeric value that is stored in two bytes. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Binary Unsigned Short | The data area is a numeric value that is stored in two bytes and does not contain any sign characters. | <ul style="list-style-type: none">• Currency• Floating• Integer |
| Binary Integer | The data area is an integer representation that is stored in binary format. | <ul style="list-style-type: none">• Currency• Floating• Integer |

Currency, floating, and integer formats, continued

| Format | Data area description | Data type |
|-----------------------|--|---|
| Binary 64-bit Integer | The data area is a 64-bit integer representation that is stored in binary format. Apply this format to any COBOL COMP numbers with more than nine digits. Note: If you have more than four implied decimals when using this option, the engine issues a warning. | <ul style="list-style-type: none">• Currency• Floating |
| Binary Float | The data area is a floating decimal representation in binary format. | <ul style="list-style-type: none">• Floating |
| Binary Double | The data area is eight-byte binary data. | <ul style="list-style-type: none">• Currency• Floating |
| Locale Currency | The data area uses the currency options for each locale as set in the system settings. | <ul style="list-style-type: none">• Currency |
| Locale General Number | The data area uses the number options for each locale as set in the system settings. | <ul style="list-style-type: none">• Currency• Floating |

4.8.3 Date Data Area Input Formats

The data format tells the engine how to read the data. Backward date support ends at December 30, 1899.

For more information about variable types, see [“Types of Variables” on page 60](#).

4.8.4 Formatted Text Data Area Input Formats

The following table lists the formats available when you are mapping a formatted text variable to a data area that contains rich text format (RTF) content, plain text content, or Base-64 encoded content. You use formatted text data areas only in DLF output.

For more information about variable types, see [“Types of Variables” on page 60](#).

For more information about formatted text variables, see *Designing for LiveEditor* in the Exstream Design and Production documentation.

Formatted text formats

| Format | Data area description |
|-------------------------|--|
| Base-64 Encoded Content | The data area is Base-64 encoded content. |
| RTF Content | The data area is Rich Text Format (RTF) content. |

Formatted text formats, continued

| Format | Data area description |
|----------------------------|---|
| Plain Text Content | The data area is plain text content. |
| Composed XML (DXF) Content | The data area is composed XML (DXF) content. If you select this option, you must also select the encoding of the content. |

4.8.5 Placeholder Data Area Input Formats

The following table lists the formats available when you are mapping a placeholder variable to a data area used to specify the location of a file containing text, images, or both. These formats control how the data is encountered in the data file object, not the content referenced by the placeholder variable.

For more information about variable types, see [“Types of Variables” on page 60](#).

The ability to use placeholder data areas is available only if you have licensed the Dynamic Content Import module.

For more information about placeholder variables, see *Importing External Content* in the Exstream Design and Production documentation.

Placeholder formats

| Format | Data area description |
|----------------------|---|
| Custom | <p>The data area uses a customized formatting.</p> <p>For more information about creating a custom format, see “Creating Custom Data Area Input Formats” on page 176.</p> |
| Keep Blanks | When the engine reads the data area, it preserves all of the spaces as they appear. |
| Trim Blanks | When the engine reads the data area, it removes all of the spaces found before the first character and after the last character. |
| Trim Leading Blanks | When the engine reads the data area, it removes all of the spaces found before the first character. |
| Trim Trailing Blanks | When the engine reads the data area, it removes all of the spaces found after the last character. |
| Lower, Keep Blanks | When the engine reads the data area, it converts the string to all lowercase characters and preserves all of the spaces as they appear. |
| Lower, Trim Blanks | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found before the first character and after the last character. |
| Lower, Trim Leading | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found before the first character. |

Placeholder formats, continued

| Format | Data area description |
|-----------------------------------|---|
| Lower, Trim Trailing | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found after the last character. |
| Lower + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. |
| Lower + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. |
| Lower + Exceptions, Trim Leading | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character. |
| Lower + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found after the last character. |
| Upper, Keep Blanks | When the engine reads the data area, it converts the string to all uppercase characters and preserves all of the spaces as they appear. |
| Upper, Trim Blanks | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found before the first character and after the last character. |
| Upper, Trim Leading | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found before the first character. |
| Upper, Trim Trailing | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found after the last character. |
| Upper + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. |
| Upper + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. |
| Upper + Exceptions, Trim Leading | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character. |
| Upper + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found after the last character. |
| InitCap, Keep Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and preserves all of the spaces as they appear. |

Placeholder formats, continued

| Format | Data area description |
|-------------------------------------|--|
| InitCap, Trim Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and removes all of the spaces found before the first character and after the last character. |
| InitCap, Trim Leading | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and removes all of the spaces found before the first character. |
| InitCap, Trim Trailing | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and remove all of the spaces found after the last character. |
| InitCap + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and preserves all of the spaces as they appear. |
| InitCap + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. |
| InitCap + Exceptions, Trim Leading | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character. |
| InitCap + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found after the last character. |
| Sentence, Keep Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and preserves all of the spaces as they appear. |
| Sentence, Trim Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found before the first character and after the last character of the file. |
| Sentence, Trim Leading | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found before the first character. |
| Sentence, Trim Trailing | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found after the last character. |
| Sentence + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. |
| Sentence + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. |

Placeholder formats, continued

| Format | Data area description |
|--------------------------------------|---|
| Sentence + Exceptions, Trim Leading | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character. |
| Sentence + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found after the last character. |
| Exceptions Only, Keep Blanks | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. |
| Exceptions Only, Trim Blanks | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found before the first character and after the last character. |
| Exceptions Only, Trim Leading | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found before the first character. |
| Exceptions Only, Trim Trailing | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found after the last character. |
| Reverse, Keep Blanks | When the engine reads the data area, it reverses the string and preserves all of the spaces as they appear. |
| Reverse, Trim Blanks | When the engine reads the data area, it reverses the string and removes all of the spaces found before the first character and after the last character. |
| Reverse, Trim Leading | When the engine reads the data area, it reverses the string and removes all of the spaces found before the first character. |
| Reverse, Trim Trailing | When the engine reads the data area, it reverses the string and removes all of the spaces found after the last character. |
| Packed (Left to Right) | The engine reads the data area from the left half-byte of the first byte to the right half-byte of the last byte. |
| Packed (Right to Right) | The engine reads the data area from the right half-byte of the first byte to the right half-byte of the last byte. |
| Packed (Left to Left) | The engine reads the data area from the left half-byte of the first byte to the left half-byte of the last byte. |
| Packed (Right to Left) | The engine reads the data area from the right half-byte of the first byte to the left half-byte of the last byte. |
| General Number | The data area is Arabic numeral data. |
| Placeholder Content | The engine does not alter the data. If you select this option, you must also select the encoding of the content. |

4.8.6 String and Tagged Text Data Area Input Formats

The following table lists the formats available when you are mapping a string or tagged text variable to a data area used for text.

For more information about variable types, see [“Types of Variables” on page 60](#).

String and tagged text formats

| Format | Data area description | Data type |
|----------------------|---|--|
| Custom | <p>The data area uses a customized formatting.</p> <p>For more information about creating a custom format, see “Creating Custom Data Area Input Formats” on page 176.</p> | <ul style="list-style-type: none">• String• Tagged text |
| Keep Blanks | When the engine reads the data area, it preserves all of the spaces as they appear. | <ul style="list-style-type: none">• String• Tagged text |
| Trim Blanks | When the engine reads the data area, it removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Trim Leading Blanks | When the engine reads the data area, it removes all of the spaces found before the first character. | <ul style="list-style-type: none">• String• Tagged text |
| Trim Trailing Blanks | When the engine reads the data area, it removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Lower, Keep Blanks | When the engine reads the data area, it converts the string to all lowercase characters and preserves all of the spaces as they appear. | <ul style="list-style-type: none">• String• Tagged text |
| Lower, Trim Blanks | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Lower, Trim Leading | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found before the first character. | <ul style="list-style-type: none">• String• Tagged text |
| Lower, Trim Trailing | When the engine reads the data area, it converts the string to all lowercase characters and removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |

String and tagged text formats, continued

| Format | Data area description | Data type |
|-----------------------------------|--|---|
| Lower + Exceptions, Keep Blanks | <p>When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and preserves all of the spaces as they appear.</p> <p>For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation.</p> | <ul style="list-style-type: none"> • String • Tagged text |
| Lower + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Lower + Exceptions, Trim Leading | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |
| Lower + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string to all lowercase characters, except those words found in an exceptions file, and removes all of the spaces found after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper, Keep Blanks | When the engine reads the data area, it converts the string to all uppercase characters and preserves all of the spaces as they appear. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper, Trim Blanks | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper, Trim Leading | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper, Trim Trailing | When the engine reads the data area, it converts the string to all uppercase characters and removes all of the spaces found after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper + Exceptions, Keep Blanks | <p>When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and preserves all of the spaces as they appear.</p> <p>For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation.</p> | <ul style="list-style-type: none"> • String • Tagged text |
| Upper + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Upper + Exceptions, Trim Leading | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |

String and tagged text formats, continued

| Format | Data area description | Data type |
|-------------------------------------|--|--|
| Upper + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string to all uppercase characters, except those words found in an exceptions file, and removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap, Keep Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and preserves all of the spaces as they appear. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap, Trim Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap, Trim Leading | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and removes all of the spaces found before the first character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap, Trim Trailing | When the engine reads the data area, it converts the string so that each word begins with an uppercase character and removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap + Exceptions, Trim Leading | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character. | <ul style="list-style-type: none">• String• Tagged text |
| InitCap + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string so that each word begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Sentence, Keep Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and preserves all of the spaces as they appear. | <ul style="list-style-type: none">• String• Tagged text |
| Sentence, Trim Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found before the first character and after the last character of the file. | <ul style="list-style-type: none">• String• Tagged text |

String and tagged text formats, continued

| Format | Data area description | Data type |
|--------------------------------------|--|---|
| Sentence, Trim Leading | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |
| Sentence, Trim Trailing | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character and removes all of the spaces found after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Sentence + Exceptions, Keep Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. | <ul style="list-style-type: none"> • String • Tagged text |
| Sentence + Exceptions, Trim Blanks | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Sentence + Exceptions, Trim Leading | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |
| Sentence + Exceptions, Trim Trailing | When the engine reads the data area, it converts the string so that only the first word of a sentence begins with an uppercase character, except those words found in an exceptions file, and removes all of the spaces found after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Exceptions Only, Keep Blanks | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and preserves all of the spaces as they appear. For more information about creating an exceptions file, see <i>System Administration</i> in the Exstream Design and Production documentation. | <ul style="list-style-type: none"> • String • Tagged text |
| Exceptions Only, Trim Blanks | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none"> • String • Tagged text |
| Exceptions Only, Trim Leading | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found before the first character. | <ul style="list-style-type: none"> • String • Tagged text |
| Exceptions Only, Trim Trailing | When the engine reads the data area, it converts the words found in the exception file to match their format in the file and removes all of the spaces found after the last character. | <ul style="list-style-type: none"> • String • Tagged text |

String and tagged text formats, continued

| Format | Data area description | Data type |
|-------------------------|--|--|
| Reverse, Keep Blanks | When the engine reads the data area, it reverses the string and preserves all of the spaces as they appear. | <ul style="list-style-type: none">• String• Tagged text |
| Reverse, Trim Blanks | When the engine reads the data area, it reverses the string and removes all of the spaces found before the first character and after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Reverse, Trim Leading | When the engine reads the data area, it reverses the string and removes all of the spaces found before the first character. | <ul style="list-style-type: none">• String• Tagged text |
| Reverse, Trim Trailing | When the engine reads the data area, it reverses the string and removes all of the spaces found after the last character. | <ul style="list-style-type: none">• String• Tagged text |
| Packed (Left to Right) | The engine reads the data area from the left half-byte of the first byte to the right half-byte of the last byte. | <ul style="list-style-type: none">• String• Tagged text |
| Packed (Right to Right) | The engine reads the data area from the right half-byte of the first byte to the right half-byte of the last byte. | <ul style="list-style-type: none">• String• Tagged text |
| Packed (Left to Left) | The engine reads the data area from the left half-byte of the first byte to the left half-byte of the last byte. | <ul style="list-style-type: none">• String• Tagged text |
| Packed (Right to Left) | The engine reads the data area from the right half-byte of the first byte to the left half-byte of the last byte. | <ul style="list-style-type: none">• String• Tagged text |
| Zoned | The data area is a decimal or a currency formatted field on a mainframe system. | <ul style="list-style-type: none">• String• Tagged text |

4.8.7 Creating Custom Data Area Input Formats

If the format you want is not available, you can create a custom format.

To create a custom data area input format:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a mapped data area and select **Data Area > Data Area Properties**.

The **Data Area Properties** dialog box opens.

3. From the **Format** list, select **Custom**.

The box below the list becomes active.

4. Right-click the box and select **Format**.

The **Select Formatting String** dialog box opens.

5. Use the formatting strings to build a customized variable.

6. Click **OK**.

The **Select Formatting String** dialog box closes.

7. Click **OK**.

The **Data Area Properties** dialog box closes.

8. From the **Edit** menu, select **Save**.

Chapter 5: Setting Up XML Formatted Data Files

In addition to the other data file formats Exstream Design and Production supports, it also supports data file formats that use XML to define the structure of the data. Increasingly, XML is the format of choice for integrating Exstream Design and Production with other systems. The XML functionality of Exstream Design and Production lets you leverage existing infrastructure using a web service, or leverage data from other sources such as PDF forms or Live documents.

Setting up an XML formatted data file, no matter the type or format, is similar to setting up a data file in other formats. This chapter discusses the following topics:

- “[Module Requirements for XML Data Source Formats](#)” below
- “[Creating XML Formatted Data Files](#)” on the next page
- “[Creating Schema Model Data Files](#)” on page 215

5.1 Module Requirements for XML Data Source Formats

To use the XML formatted data source formats that are available in Exstream, you must license the modules that are related to your specific business requirements. The following table describes the data source formats that use XML to structure data and the modules that are required for each data source format.

XML formatted data file format module requirements

| Data file format | Required module | Related information |
|---|--|---|
| Interactive | InteractiveInput | <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation |
| PDF Form | <ul style="list-style-type: none">• PDF Form Miner• PDF Form Pre-Fill | “Setting Up Data Files to Pre-Fill and Mine PDF Forms” on page 287 |
| Schema model | XML/JSON Input | “Creating Schema Model Data Files” on page 215 |
| Web Service (RESTful) or Web Service (SOAP) | Web Services Interface | “Setting Up Data Files to Communicate with a Web Service” on page 272 |
| XML data file (as input) | XML/JSON Input | “Creating XML Formatted Data Files” on the next page |
| XML data file (as output) | XML/JSON (Data) Output | “Setting Up XML Formatted Data Files” above |

For more information about Exstream Design and Production modules, see *Getting Started* in the Exstream Design and Production documentation.

5.2 Creating XML Formatted Data Files

Creating an XML formatted data file, no matter the type, is similar to creating a data file in other formats. First, you create a data file object that defines the data file type and format. Then, you use the data file properties to define the general properties that apply to all of the data in the data file.

To create an XML formatted data file, you must complete the following tasks:

1. [“Considerations for Setting Up an XML Formatted Data Source” below](#)
2. [“Creating a Data File Object” on page 255](#)
3. [“Defining the Data Mapping Method” on page 256](#)
4. [“Selecting the Data File Sources” on page 257](#)
5. [“Mapping an XML Formatted Data File” on page 189](#)

You can also complete the following optional tasks as needed:

- [“Validating a Data Source at Testing and Run Time Using a Schema” on page 232](#)
- [“Transforming Data at Testing and Run Time” on page 235](#)
- [“Using Attributes to Identify the XML Hierarchy” on page 208](#)
- [“Defining the Structure of an XML Formatted Output Data File” on page 209](#)
- [“Using a Variable Report to Create a Sample XML Data Layout” on page 210](#)
- [“Integrating XML Schemas with XML Formatted Data Files” on page 212](#)

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

5.2.1 Considerations for Setting Up an XML Formatted Data Source

As with the other data file formats, you use an XML formatted data file to define and point to existing data sources that are created and stored outside of Exstream Design and Production. The structure defines how you create and map an XML formatted data file. You use the structure to define the start of customers and sections and to identify the areas in which the engine can find data. As shown in the following graphic, the different parts of the XML structure work together to define the layout of data.

XML formatted data file in the Edit Panel

```
<?xml version="1.0"?>
<BankStatements>
    <Init>
        </Init>
    1   <Customer Account="4172130">
        <FirstName>Thomas</FirstName>
        <LastName>Reagan</LastName>
        <Email>tjreagan@email.com</Email>
        <City>Savannah</City>
        <State>GA</State>
        <Zip>30077</Zip>
        <Transactions>30</Transactions>
        <Bank1>Cust_B1</Bank1>
        <Savings>
            2   <Debit>$100.01</Debit>
            <Debit>$200.02</Debit>
            <Credit>$300.03</Credit>
            <Credit>$400.04</Credit>
        3   </Savings>
        4   <recipientStart>
            <profile>Recipient Copy 2</profile>
            <firstName>John</firstName>
            <lastName>Kohlby</lastName>
            <spaceTag>Platinum</spaceTag>
            <team>blue</team>
        5   </recipientStart>
            <Bank2>Cust_B2</Bank2>
            <Savings>
                <Debit>$500.05</Debit>
                <Debit>$600.06</Debit>
                <Credit>$700.07</Credit>
                <Credit>$800.08</Credit>
            </Savings>
            <Bank5>Cust_B5</Bank5>
        </Customer>
    </BankStatements>
```

| | |
|---|--------------------------|
| 1 | Customer start indicator |
| 2 | Section start indicator |
| 3 | Section end indicator |

| | |
|---|--------------------------------|
| 4 | Recipient start indicator |
| 5 | Recipient end indicator |
| 6 | Attribute value |
| 7 | Tag name |
| 8 | Unmapped tag value (data area) |
| 9 | Mapped tag value (data area) |

Tip: In addition to using indicators to identify the different parts of the XML structure, Exstream also uses different colors in the Edit panel to indicate data file mappings and data behaviors.

To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select **View > What do colors mean**.

To help you set up your XML formatted data source, keep in mind the following considerations::

| Data Source Consideration | Description |
|-----------------------------|---|
| Data structure requirements | <p>XML formatted data files used with Exstream Design and Production must be well-formed XML documents that comply with the W3C XML 1.0 specification. However, there is one limitation: you cannot have any text that comes after a child tag. The following sample compares those structures that are supported in Exstream Design and Production with a structure that is not supported in Exstream Design and Production. Keep in mind that only one sample record is required for data mapping. Also, do not include blank tags that contain no tag values; instead, use a character such as x to represent data areas.</p> <p>Sample supported and unsupported XML structures</p> <div style="border: 1px solid black; padding: 10px;"><p>Supported</p><p>a) <a>lorem ipsum</p><p>b) <a>lorem ipsum<c>Dolor</c></p><p>c) <a>Dolor lorem ipsum</p><p>Unsupported</p><p><a>lorem ipsum Dolor</p></div> <p>The hierarchy of an XML formatted data file and the time at which the engine reads data affect each other. The engine reads or writes an XML data file in the order it encounters the data. By default, it ignores tag hierarchy and does not maintain relationships between the levels in the data. You can structure your data in such a way, or select how and when the engine reads data, in order to create the output you require.</p> <p>If you must ensure that a file is valid, in addition to being well-formed, you can use the <code>Driver.xsd</code> schema, which is available in your installation directory, or a custom schema to structure an XML formatted data file. Schemas define the structure of the building blocks that make up an XML file. They include information such as the available elements and attributes, the data types for the elements and attributes, and the default values for the elements and attributes. Data types are especially important in Exstream Design and Production since you can use the data type information in the schema to create variables and define the format of the data. In addition, you can use a schema to validate data coming from a data source and to create transforms that let you create XML data files with the exact structure you require.</p> <p>For more information about using schemas, see "Integrating XML Schemas with XML Formatted Data Files" on page 212.</p> |

| Data Source Consideration | Description |
|---|--|
| Encoding support | <p>Exstream Design and Production has several encoding options that you can use with DBCS XML formatted data files. UTF-8 encoding is also available as a character set in SBCS XML formatted data files, in addition to DBCS XML formatted data files.</p> <p>For more information about selecting an encoding or a character set, see "Defining the Records in a Customer Driver File" on page 25.</p> <p>If you select the UTF-8 encoding for an XML formatted data file, you must meet the following requirements:</p> <ul style="list-style-type: none">• You must be able to correctly view the characters using the active Windows code page. During packaging, Design Manager automatically includes this code page, and the engine uses it to map the UTF-8-encoded characters. If you use a code page other than Windows 1252, you must select the correct font and script in order to view the characters in the Edit Panel.• If you are creating an SBCS XML formatted data file and the active code page is DBCS, Exstream Design and Production supports only characters with values 0 through 127. To use all the characters in a DBCS code page correctly, create a DBCS XML formatted data file.• If you include the encoding attribute in the declaration tag, it must be <code>encoding="UTF-8"</code>. Keep in mind, though, that the encoding attribute is optional.• Exstream Design and Production does not create Byte Order Markers (BOMs) in output XML formatted data files. BOMs are not necessary, but you can include them in input XML formatted data files. |
| Namespace support | <p>XML formatted data sources can include multiple namespaces. Keep in mind that if you created an XML formatted data file in a version of Exstream Design and Production earlier than 8.0.301, and you now want to use namespaces, you must manually enable namespace support. In the data file properties, on the Advanced tab, select the Enable XML namespace support check box. This check box is selected by default when you create new XML formatted data files.</p> |
| Transferring a data source between platforms | <p>If you transfer an XML formatted data source between platforms (such as z/OS, Windows NT, and UNIX), view the data in the new platform to verify that the file transferred correctly. When you transfer an XML formatted data source, the line endings might change if you transfer in ASCII format. This change can adversely affect the way Exstream Design and Production parses the XML formatted data source and can cause run-time errors if the ends of lines are incorrectly presented on the new platform.</p> |
| Using XML data sources to create report files | <p>If you are creating a report file, it is possible for the data to appear in an order you do not expect. For example, if a parent tag has a timing different from its child tag, the parent tag could appear in the report below the child tag. To make sure a report file includes data in the order you expect, arrange the tags in the data source in a way that lets you map the variables in the same order that the engine populates the variables during production. If the report is at the application level, use tags to mark customer data. If the report is an output queue-level report, however, use record properties.</p> <p>For more information about creating a report file, see "Creating a Report or a Post-sort Report File" on page 50.</p> |

| Data Source Consideration | Description |
|-------------------------------|---|
| Eliminating redundant content | <p>If it is possible for a design to have redundant content, you can use the data aggregation process to eliminate the redundancy in complex documents, such as insurance policies and publications. Using data aggregation to eliminate redundancy is available only if you use an XML customer driver file that contains data sections.</p> <p>During the data aggregation engine timing, the engine has access to all of the customer data at one time. This engine timing allows the engine to read all of the data, including any section data, and to hold the data in memory until it reaches the next customer.</p> <p>When you use the data aggregation process to eliminate repetition, the engine then merges content as necessary using the data values in the data file, the settings in the design, and the variables that control how the engine reads, manipulates, and deletes duplicate section data.</p> <p>For more information about configuring a design for data aggregation, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> <p>Because the engine reads all of the data for a customer during data aggregation, you can also use data aggregation to change how Exstream Design and Production reads the hierarchy in an XML formatted data file. Some XML input files require a specific data hierarchy to define relationships between data. For example, suppose that you have an XML data file that has three levels of data. By default, the engine does not recognize that a third level tag is a child of a second level tag. Restructuring the data file can require the addition of redundant data sections. Instead of restructuring, use the AGGREGATE_DATA engine switch to enable data aggregation. The AGGREGATE_DATA engine switch requires additional memory and processing time, but it lets you use the original data file hierarchy.</p> <p>During data aggregation, the engine does the following:</p> <ul style="list-style-type: none">• Reads all of the customer data, including data sections• Compares data values• Changes and deletes content based on aggregation settings on variables and data values• Triggers rules on variables with the Compute time option set to Data aggregation <p>For more information about the AGGREGATE_DATA engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |

5.2.2 Creating a Data File Object

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. Select the type and format of the data file you want to create.
4. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

For more information about which data file types are available for each format, see “[Data File Type and Format Availability](#)” on page 14.

5.2.3 Defining the Data Mapping Method

One step in creating an XML or a JSON formatted data file is selecting how the data file will be mapped. The data mapping method determines how the variables are created as part of the data mapping process and the amount of customization that is available when you map.

For more information about mapping an XML or JSON formatted data file, see the following topics:

- “[Mapping an XML Formatted Data File](#)” on page 189
- “[Mapping a JSON Formatted Data File](#)” on page 260

To define the data mapping method:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To specify the method for data mapping, you must complete one of the following sets of steps:

| To | Do this |
|--|---|
| Automatically map the data file using a predefined schema or a schema contained in a Web Services Description Language (WSDL) file | <p>From the Data mapping method list, select Schema.</p> <p>Note: This option is available for XML data files, input Live data files, Web Service (RESTful) data files, and Web Service (SOAP) data files only. You cannot use a schema to map JSON data files, output Live data files or PDF form data files.</p> |
| Manually or automatically map the data file in the Edit Panel | <p>From the Data mapping method list, select Sample file (manual).</p> |
| Dynamically map the data file using a specified list of variable names that match the tag names | <ol style="list-style-type: none">a. From the Data mapping method list, select Sample file (simple). If you select this option, you cannot customize the data area format. Instead, the engine automatically uses the default settings. The engine also ignores data sections.b. In the Top level tag box, enter the top level tag. This tag is the start of the data file.c. In the Customer tag box, enter the tag used to indicate a new customer.d. In the Variables for simple mapping box, enter the variables used to map the XML data file. <p>Note: This option is not available for JSON data files.</p> |

| To | Do this |
|---|---|
| Dynamically map all of the tags in a data file using variable names that match the tag name | <ol style="list-style-type: none">From the Data mapping method list, select Sample file (dynamic). If you select this option, you cannot customize the data area format. Instead, the engine automatically uses the default settings. The engine also ignores data sections.In the Top level tag box, enter the top level tag. This tag is the start of the data file.In the Customer tag box, enter the tag used to indicate a new customer. <p>Note: This option is not available for JSON data files.</p> |

4. If you selected either **Sample file (simple)** or **Sample file (dynamic)** from the **Data mapping** list and are mapping a data file that contains at least one array, complete one of the following sets of steps, depending on how the array is defined:

| If the array is defined in this way | Do this |
|---|--|
| A group tag that matches the array variable name encloses the data elements. All of the data elements use the same tag. | <ol style="list-style-type: none">From the Array variable name handling list, select Use variable name for group.In the Array element tag box, enter the tag name for each data element in the array variable. |
| A group tag that matches the array variable name followed by a suffix encloses the data elements. The tag for each of the data elements matches the name of the array variable. | <ol style="list-style-type: none">From the Array variable name handling list, select Use variable name for element.In the Array group tag suffix box, enter the suffix that follows the variable name in the tag that introduces the group of data elements. For example, if you enter <code>_Array</code> and the name of the variable is <code>Debit</code>, the tag will be <code><Debit_Array></code>. |
| A single tag encloses all of the data elements for multiple array variables. The tag for each of the data elements matches the name of the array variable. | <ol style="list-style-type: none">From the Array variable name handling list, select Use for element, single group.In the Array group tag box, enter the name of the tag that encloses all of the data elements for all of the array variables. |

5. From the **Edit** menu, select **Save**.

5.2.4 Selecting the Data File Sources

When you select the data source for an XML or a JSON formatted data file, you define the location of the data you use for mapping, testing, and production.

A mapping data source is typically the simplest of the data sources you use. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. If you are mapping an XML or an input Live data file (for example,

customer driver files or reference files), you can use a schema as the mapping source. This option is not available for JSON data files.

A test data source is typically a subset of the production data that you use to test and troubleshoot an application before putting it into production. To ensure that your tests are as accurate as possible, the test data source and the production data source should be similar. During a test, the engine uses the data maps that you created using the mapping data source, along with the test data source, to read and write customer data. You can then use the results to make sure that the final application design is accurate.

A production data source is the complete set of data required to produce the output you send to customers. During production, the engine uses the data maps that you created using the mapping data source, along with the production data source, to read and write customer data.

For more information about mapping a data file, see [“Data File Mapping” on page 129](#).

To select the data file sources:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To specify the location of the data source to use during mapping, complete one of the following sets of steps:

| To | Do this |
|---|--|
| Use a sample data file | <ol style="list-style-type: none">a. From the Data mapping method list, select Sample file (manual).b. In the Data mapping source box, enter the file path to the external file used to provide data. |
| Use a schema or a schema contained in a Web Services Description Language (WSDL) file | <ol style="list-style-type: none">a. From the Data mapping method list, select Schema.b. In the Data mapping source box, select a schema that defines the layout of the data or a WSDL file that contains a schema that defines the layout of the data.c. In the Root element list, select the root element that selects the portion of the schema that you want to use. The options vary based on the root elements in the schema. The root element also determines which variables are available when you map. |

Note: This option is available for XML data files, input Live data files, XML formatted Web Service (RESTful) data files, and XML formatted Web Service (SOAP) data files only. You cannot use a schema to map output data files, JSON data files, Live data files or PDF form data files.

4. Click the **Advanced** tab.
5. If data for a customer can change during processing (if you are using a connector to connect to the data source, for example) and you want to update the information in the data source, complete the following steps:

- a. Select the **File can be updated** check box.
- b. In the **Update output file name when running in test mode** box, enter the path to the output file.
- c. In the **Update output file name when running in production mode** box, enter the path to the output file.

Note: Use the Trigger function to control the timing of the update operation.

For more information about the Trigger function, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

6. If you are using XML data files, and the files you use include namespaces, make sure that the **Enable XML namespace support** check box is selected.
7. In the **Rule** box, enter a rule to include or exclude the file during an engine run.
8. On the **Test Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use for testing and, optionally, mapping:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in test box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use in test box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

9. On the **Production Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in production box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

10. From the **Edit** menu, select **Save**.

5.2.5 Mapping an XML Formatted Data File

You can map an XML formatted data file in the Edit Panel using the XML automapping feature, or you can map the data file manually. Automapping lets you automatically find areas of a data file, identify the type of data in the area, and map a variable. Manual mapping includes tasks that you must complete for all data files, such as defining the start of a customer, whether the data files are automapped or manually mapped. If you must map variables with names that do not match the tag names, you must use manual mapping. You also use manual mapping when you must map attribute values in addition to the tag values.

After you map a data area, Design Manager highlights the data area in a different color. If you want information about what the highlighting colors mean in the Edit Panel, right-click the Edit Panel and select **View > What do colors mean**.

Keep in mind that you are not required to map variables to all of the tags in a data file. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory required, map only those tags you want to use to populate variables.

To map an XML formatted data file, complete the following tasks as needed:

- “[Automapping an XML Formatted Data File](#)” on the next page
- “[Manually Mapping an XML Formatted Data File](#)” on page 193

- “[Viewing the Data Mapping Layout](#)” on page 271

Automapping an XML Formatted Data File

You can use automapping to automatically map tag values or attribute values. If you have XML data prepared for mapping and have not yet created variables, you can also create variables based on information in tags or in a schema. If you are creating variables during automapping, Design Manager reads the file or the schema and selects the basic name, data type, and format features of a variable based on the data source. Although minor modifications might be required to change the default selection for a particular tag, or to define properties such as valid values or output format, automapping is usually faster than manual mapping. After variables are created, or if you are using existing variables, Design Manager maps them to the data automatically.

You can use the automapping feature to map the data file using new variables created as part of the automapping process, or using existing variables. If you are creating new variables, Design Manager uses the tag names and attributes, or the information in the schema, to name the variables and to define the properties on the **XML Auto-Map** dialog box. All other variable and data area properties are set to the default. Keep in mind that schemas can define data types that do not match data types in Exstream Design and Production. In this situation, a data type supported by Exstream Design and Production is substituted when Exstream Design and Production creates the variables. For all automatically created variables, make sure that you verify all of the variable and data area properties. If you want to use existing variables to automap an XML formatted data file, the variable names and properties must match the variable names and properties specified in the **XML Auto-Map** dialog box.

If you are using a schema to automap an XML data file and you do not have mapping data, you can use the schema to create variables automatically. Then, you can create a sample data file and map the variables to data. The variables you map to the data file must match the variables created using the schema and the variables used in the design. The variables you create using a schema are the same as the variables you create manually or those you create as you map a data file. You can use these schema-created variables alone or in combination in a design just as you would other variables.

For more information about creating variables automatically from XML or an XML schema separately from the automapping process, see “[Creating Variables Automatically from XML or an XML Schema](#)” on page 74.

To automap an XML formatted data file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. From the **Data mapping method** list, select **Sample file (manual)** or **Schema**.
4. From the **File** menu, select **Save**.

5. Depending on when you want to complete the automapping, complete one of the following sets of steps to automap the data file:

| To | Do this |
|---|--|
| Automap the data file during data file creation | On the Basic tab, click  . The XML Auto-Map dialog box opens. |
| Automap the data file after it has been created | a. From the Library, drag the data file to the Edit Panel. b. Right-click any empty area and select Tag > Auto-Map . The XML Auto-Map dialog box opens. The XML Auto-Map dialog box opens. |

6. In the **Folder** box, select the folder in the Library in which to store new variables or to find existing variables.
7. If you have one or more data areas with existing maps and you want to remove the maps before automapping, select the **Remove all existing data maps** check box. Clear the check box if you do not have existing data maps or if you want to double-map the data areas with existing maps.
8. To name the variable, do one or more of the following:

| To | Do this |
|---|--|
| Change the name of the variable | In the Name box, enter the name you want to use for the variable. By default, the tag name or the attribute value defines the variable name. If the tag name or the attribute value has a space, Design Manager substitutes an underscore for the space. |
| Add a prefix to all variable names | a. In the Variable name prefix box, enter a string of characters. b. Click Apply . |
| Add the parent tag name to resolve duplicate variable names | Click Resolve using parent names . Note: If you have duplicate tag names, keep in mind that the order in which you add the prefix affects the final variable name. If you resolve the duplicate names using parent names before you add a prefix, Design Manager adds the prefix in front of the parent name. However, if you add the prefix before resolving the duplicates, Design Manager adds the parent name in front of the prefix. |

9. To update the properties of a variable, complete the following steps:
 - a. Click the tag in the hierarchy tree area.
 - b. From the **Type** list, select a data type.
 - c. If the variable is an array, select the **Array** check box. Design Manager does not

detect whether a variable should be an array. If the variable can have more than one value for a customer, you must select the **Array** check box.

- d. From the **Format** list, select an output format. The options vary depending on your selection from the **Type** list.

For more information about variable types and output formats, see “[Types of Variables](#)” on [page 60](#) and “[Formatting Variable Values in Output](#)” on [page 81](#).

10. To exclude specific tags from automapping, complete one of the following sets of steps:

| To | Do this |
|---|---|
| Exclude a specific tag from automapping | <ol style="list-style-type: none">a. Click the tag in the hierarchy tree area.b. Select the Ignore check box. |
| Exclude all tags except the selected tag and all its nested tags | <ol style="list-style-type: none">a. Click the parent tag in the hierarchy tree area.b. Click Ignore all but this branch. |
| Clear ignore flags and include all of the ignored tags in automapping | Click Clear all ignore flags . |

11. If you are automapping a report file, complete the following steps to define the action to take when data is missing:

- a. To define the action you want the engine to take when it encounters a missing value, select one of the following options from the **Action on missing value** list:
 - **Output empty tag**—Include the tag in the report and include no value.
 - **SUPPRESS tag**—Do not include the tag in the report.
 - **Throw an error**—Do not include the tag in the report and issue an error message.
- b. To apply the option you selected from the **Action on missing value** list to all of the tags in the data file, click **Apply to all**.
- c. To apply the option you selected from the **Action on missing value** list if the engine encounters a zero or a blank in the tag value, select the **Interpret zero or blank tag value as missing data** check box.

If you are using the data file with a DLF (interactive document), make sure you select this check box. DLFs do not remove tags when information is removed; instead the DLF leaves the tag value blank.

- d. To interpret zero or blank tag values in all of the tags in the data file as missing data, click **All**.

12. Click **OK**.

Design Manager automaps the data file and highlights the data areas based on their use to indicate that you have mapped values to the data areas. In addition, you can view information (such as the name) about the variable when you place your pointer over the data area.

Manually Mapping an XML Formatted Data File

You can manually map all of the XML formatted data file types. Manual mapping is especially useful when you must use variable names that are different from the XML tag names or when you want to customize the options that are set during automapping.

Note: The tasks in this section assume that you are mapping any XML formatted data file type, with the exception of schema model data files. For information about mapping schema model data files, see “[Manually Mapping a Schema Model Data File](#)” on page 222.

The tasks in this section also assume that you have selected **Sample file (manual)** or **Schema** from the **Data mapping method** list on the **Basic** tab of the XML data file properties. These data mapping methods let you manually map the XML data file.

If the **Data mapping method** list is set to **Sample file (dynamic)** or **Sample file (simple)**, the XML data file is intended to be automapped.

For more information about automatically mapping a data file, see “[Automapping an XML Formatted Data File](#)” on page 190.

This section discusses the following topics:

- “[Identifying Customers in XML Data](#)” below
- “[Identifying Data Sections in XML Data](#)” on the next page
- “[Identifying Recipients in XML Data](#)” on page 195
- “[Manually Mapping a Variable to a Tag Name, Value, or Attribute Value in XML Data](#)” on page 196
- “[Specifying When the Engine Writes Data to a Tag in a Report File](#)” on page 201

Identifying Customers in XML Data

Note: This task assumes that you are mapping any XML formatted data file type, with the exception of schema model data files. For information about mapping schema model data files, see “[Manually Mapping a Schema Model Data File](#)” on page 222.

For XML data sources, you must specify which XML element starts a new customer. You must identify at least one XML element as a customer in the data; otherwise, the engine will not process your application.

To identify a customer in XML data:

1. In Design Manager, from the Library, drag the XML data file to the Edit Panel.
2. In the data, select the unmapped customer tag name or the value that represents the customer.
3. Right-click the highlighted data and select **Tag > Properties**.

The **XML Tag Mapping Properties** dialog box opens.

4. Click the **Options** tab.
5. To select where the new customer mapping begins in the data, do one of the following:

| To | Do this |
|--|---|
| Start a new customer before the selected tag | From the Starts customer list, select At start of tag . |
| Start a new customer after the selected tag | From the Start customer list, select At end of tag . |

NOTE: By default, the **Starts customer** list is set to **None**. This setting indicates that the selected tag does not start a customer.

6. Click **OK**.

The **XML Tag Mapping Properties** dialog box closes and a solid black line appears above each instance of the tag to indicate the start of a new customer. Each customer ends when the engine encounters the next customer tag in the XML data.

7. From the Menu bar, select **Edit > Save**.

Identifying Data Sections in XML Data

For XML data sources, you can specify which XML element starts a new data section. If you do not identify the start data section tag, the engine does not reset the variables, which might cause incorrect results at run time.

For more information about section data, see [“Section Data in Exstream Design and Production” on page 4](#).

To identify a data section in XML data:

1. In Design Manager, from the Library, drag the data file to the Edit Panel.
2. In the data, select the unmapped data section tag name or select the data section tag value.
3. Right-click the highlighted data and select **Tag > Properties**.

The **XML Tag Mapping Properties** dialog box opens.

4. Click the **Options** tab.
5. To specify where the new data section mapping begins in the data, do one of the following:

| To | Do this |
|---|---|
| Always start a new data section at the selected tag | <ol style="list-style-type: none">a. From the Starts section list, select Always.b. In the Section box, enter the name of the data section. |
| Start a new data section only at the first occurrence of the selected tag | <ol style="list-style-type: none">a. From the Starts section list, select First.b. In the Section box, enter the name of the data section. |

NOTE: By default, the **Starts section** list is set to **None**. This setting indicates that the selected tag does not start a data section.

6. Click **OK**.

The **XML Tag Mapping Properties** dialog box closes and a solid red line appears above each instance of the data section tag to indicate the start of a new data section.

Note: By default, the **End section(s) or recipient(s) at closing tag** check box on the **Advanced** tab of the XML data file properties is selected. This setting indicates that data sections and recipients end at the end tag for the data section or recipient.

For best results, clear the **End section(s) or recipient(s) at closing tag** check box only for backward compatibility with Exstream version 9.0 and earlier.

For more information about upgrade considerations for data sections and recipients, see *Installation and Upgrade Information* in the Exstream Design and Production documentation.

7. From the Menu bar, select **Edit > Save**.

Identifying Recipients in XML Data

When you are using recipient data (data for recipients of additional copies of documents), you must identify where a new recipient starts. If you do not identify the start recipient tag when using recipient data, the engine does not reset the variables, which might cause incorrect results at run time.

For more information about adding recipient copies of documents, see *Designing Customer Communications* in the Exstream Design and Production documentation.

To identify a recipient in the XML data:

1. In Design Manager, from the Library, drag the data file to the Edit Panel.
 2. In the data, select the unmapped recipient tag name or select the recipient tag value.
 3. Right-click the highlighted data and select **Tag > Properties**.
- The **XML Tag Mapping Properties** dialog box opens.
4. Click the **Options** tab.
 5. To start a new recipient at the selected tag, select **Always** from the **Starts recipient** list.

NOTE: By default, the **Starts recipient** list is set to **None**. This setting indicates that the selected tag does not start a recipient.

6. Click **OK**.

The **XML Tag Mapping Properties** dialog box closes. A solid green line appears above each instance of the recipient tag to indicate the start of a new recipient.

Note: By default, the **End section(s) or recipient(s) at closing tag** check box on the **Advanced** tab of the XML data file properties is selected. This setting indicates that data sections and recipients end at the end tag for the data section or recipient.

For best results, clear the **End section(s) or recipient(s) at closing tag** check box only for backward compatibility with Exstream version 9.0 and earlier.

For more information about upgrade considerations for data sections and recipients, see *Installation and Upgrade Information* in the Exstream Design and Production documentation.

7. From the Menu bar, select **Edit > Save**.

Manually Mapping a Variable to a Tag Name, Value, or Attribute Value in XML Data

Note: The task in this section assumes that you are mapping any XML formatted data file type, with the exception of schema model data files. For information about mapping schema model data files, see “[Manually Mapping a Schema Model Data File](#)” on page 222.

1. In Design Manager, from the Library, drag the XML data file to the Edit Panel.
2. In the data, select any unmapped tag name, value, or attribute value.
3. Right-click the highlighted data and select **Tag > Properties**.

The **XML Tag Mapping Properties** dialog box opens.

4. Click the **Basic** tab.
5. To map the tag, do one of the following:

| To | Do this |
|--|---|
| Map an existing variable to the tag name | <ol style="list-style-type: none">a. Click the Data mapped to tag name box. A prompt opens, asking whether you want to use an existing variable.b. Click Yes. The Select Variable dialog box opens.c. Select the variable from the list.d. Click OK. The Data Area Properties dialog box opens. |
| Map a new variable to the tag name | <ol style="list-style-type: none">a. Click the Data mapped to tag name box. A prompt opens, asking whether you want to use an existing variable.b. Click No. The New Variable dialog box opens.c. In the Name box, enter a name. In the Description box, enter a description (optional).d. From the Type list, select the type of data that the variable represents.e. If the variable can contain more than one value, select the Array check box.f. In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples.g. Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Map an existing variable to a tag value | <ol style="list-style-type: none">a. Click the Data mapped to tag value box. A prompt opens, asking whether you want to use an existing variable.b. Click Yes. The Select Variable dialog box opens.c. Select the variable from the list.d. Click OK. The Data Area Properties dialog box opens. |

| To | Do this |
|---|--|
| Map a new variable to a tag value | <ol style="list-style-type: none">a. Click the Data mapped to tag value box. A prompt opens, asking whether you want to use an existing variable.b. Click No. The New Variable dialog box opens.c. In the Name box, enter a name. In the Description box, enter a description (optional).d. From the Type list, select the type of data that the variable represents.e. If the variable can contain more than one value, select the Array check box.f. In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples.g. Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Map an existing variable to the attribute value | <ol style="list-style-type: none">a. For all data file types other than report, enter the attribute value in the Value box. By default, the attribute value in the tag populates the box.b. For all data file types other than report, select the The attribute/value pair is required for the mapping check box to specify that the exact value must always be found within the tag for this mapping to apply.c. Click the Data mapped to value box. A prompt opens, asking whether you want to use an existing variable.d. Click Yes. The Select Variable dialog box opens.e. Select the variable from the list.f. Click OK. The Data Area Properties dialog box opens. |

| To | Do this |
|---|--|
| Map a new variable to the attribute value | <ul style="list-style-type: none"> a. For all data file types other than report, enter the attribute value in the Value box. By default, the attribute value in the tag populates the box. b. For all data file types other than report, select the The attribute/value pair is required for the mapping check box to specify that the exact value must always be found within the tag in order for this mapping to apply. c. Click the Data mapped to value box. A prompt opens, asking whether you want to use an existing variable. d. Click No. The New Variable dialog box opens. e. In the Name box, enter a name. In the Description box, enter a description (optional). f. From the Type list, select the type of data that the variable represents. g. If the variable can contain more than one value, select the Array check box. h. In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples . i. Click Finish. <p>The New Variable dialog box closes and the Data Area Properties dialog box opens.</p> |
| Skip the selected tag during processing | <p>Select the Ignore this tag check box. If you select this option, all of the other options on the dialog box are disabled. Skip to step 8.</p> |

- In the **Data Area Properties** dialog box, verify or update the properties for the data area.

For information about specifying how the engine uses a data area during processing, see [“Specifying How the Engine Uses the Data Area During Processing” on page 314](#).

For information about specifying the input format of data, see [“Specifying the Input Format of Data in a Data Area” on page 154](#).

- Click **OK**.

The **Data Area Properties** dialog box closes.

- On the **XML Tag Mapping Properties** dialog box, click the **Options** tab.
- To specify how the engine processes the data, complete the following steps:
 - To remove white space at the beginning or end of the tag value, select the **Suppress whitespace** check box.
 - To specify that the engine should apply a default value defined in the variable if a tag is missing, select the **Use previous value if missing** check box.
- If you are creating an auxiliary layout file, a report file, or a post-sort report file, you must complete the following steps to specify how the engine writes out data from the data file:

- a. To define the action that you want the engine to take when it encounters a missing value, complete the following steps:
 - i. From the **Action on missing value** list, select one of the following options:
 - **Output empty tag**—Include the tag in the output file and include no value.
 - **Suppress tag**—Do not include the tag in the output file.
 - **Throw an error**—Do not include the tag in the output file and issue an error message.
 - ii. If the engine should identify a zero or a blank tag value as a missing value, select the **Interpret zero or blank tag value as missing data** check box.

The action that you selected from the **Action on missing value** list is applied to any zero or blank value in the data.

Note: If you are using the data file with DLF output, make sure that you select the **Interpret zero or blank tag value as missing data** check box. DLF files do not remove tags when information is removed; instead the DLF file leaves the tag value blank.

- b. If you are mapping an array variable, in the **Multiplicity for arrays** area, complete the following steps as needed to specify how array elements increment missing tags with default values to keep arrays in sync within the XML hierarchy:

| To | Do this |
|---|--|
| Prevent arrays from incrementing missing tags | In the Min value box, enter 0 . |
| Specify the minimum number of times to increment the tag under the parent tag | In the Min value box, enter the minimum number of times to repeat the tag. |
| Specify a maximum number of times to increment the tag under the parent tag | <ol style="list-style-type: none">i. Clear the Unbounded max value check box.ii. In the Max value box, enter the maximum value. |
| Specify that there is no maximum number of times to increment the tag | Select the Unbounded max value check box. |

- c. If you are mapping a report file and want to filter the output using a rule, enter a rule in the **Filter** box.

11. Click **OK**.

The **XML Tag Mapping Properties** dialog box closes and Design Manager applies a color to the tag value to indicate that you have mapped the name to a variable. The color of the data mapping can vary depending on the settings that you apply. To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select

View > What do colors mean. In addition, you can also view information about the mapping (such as the variable name) when you hover the pointer over a mapped data area.

For more information about data area properties, see “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151.

Specifying When the Engine Writes Data to a Tag in a Report File

You can change the engine timing so that the engine writes data to a report file or to a post-sort report file at different times. You can define the engine timing either in the data file properties or in the tag properties. When you want to apply the same timing to all of the tags in the data file, define the engine timing in the data file. When you want to apply specific timing to individual tags, define the engine timing in the properties for each tag.

For information about defining the engine timing in a report or post-sort report file, see “[Defining the Records in a Report or Post-sort Report File](#)” on page 55.

To specify when the engine writes data to a tag in a report file:

1. In Design Manager, from the Library, drag a report file to the Property Panel.
2. Click the **Advanced** tab.
3. From the **IO time** list, select **As determined by record properties**.
4. From the **File** menu, select **Save**.
5. From the Library, drag the report file to the Edit Panel.
6. Highlight any unmapped tag name or value, right-click, and select **Tag > Properties**.

The **XML Tag Mapping Properties** dialog box opens.

7. Click the **Basic** tab.
8. Map a tag name, value, or attribute value.
9. Click the **Options** tab.
10. From the **IO time** list, select an option to specify when the engine reads data records.
11. Click **OK**.

The **XML Tag Mapping Properties** dialog box closes and Design Manager highlights the tag value, based on its use, to indicate that you have mapped the name to a variable. In addition, you can view information about the variable (such as the name) when you place your pointer over the data area.

Viewing the Data Mapping Layout

You can view details about the data mapping layout of a data file. For example, you can review the properties of all of the variables that are mapped to the data file.

To view the data mapping layout:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click and select **View > View Layouts**.

A list of the tags or elements in the layout opens in the Property Panel.

3. Click a specific tag or element, and if it is present in the mapped data file, it is highlighted in the Edit Panel.

5.2.6 Validating a Data Source at Testing and Run Time Using a Schema

During testing and production, the engine reads data from the data source. If you are defining an input XML data file (for example, customer driver files or reference files) or a schema model data file, you can use a schema to ensure that the data source is valid and conforms to the standard that is defined by the schema. The engine can validate unstructured data before it is transformed or, if you are not transforming unstructured enterprise data, the engine can validate structured data that is sent directly to the engine. Because you can transform the data into its final structure during processing, the data is not required to be structured before it is validated.

For more information about transforming data, see “[Transforming Data at Testing and Run Time](#)” on page 235.

You can use different schemas to validate data during testing and during production, and neither schema is required to match the schema that is used to map the data file. Keep in mind that if you validate data, the engine performance might decrease during testing and production.

You can optionally include the schema that you used for testing in the package file in order to use the same schema for production, and to keep the schema with the application so that you do not have to provide access to or distribute a separate file after packaging.

Caution: To validate XML data on the z/OS platform, you must first download and install the “XML Toolkit for z/OS” from the IBM website. Then, you must point your engine to run Job Control Language (JCL) to the installed Partitioned Data Set Extended (PDSE).

If the engine finds an error, only the first instance of the first deviation from the schema is included in the engine error message file. This restriction helps to prevent the engine error message file from becoming too large. Also, keep in mind that you might receive errors if the declared encoding does not match the actual encoding of the schema, even if the data is valid.

Note: The task in this section assumes that you are using a schema model data file or that you are using an XML data file and have selected **Schema** from the **Data mapping method** list on the **Basic** tab. You must use one of these schema-compatible files in order to validate your data using a schema at testing and run time.

To validate the data source at testing and run time using a schema:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To validate the test data source, complete the following steps:
 - a. Click the **Test Data Source** tab.
 - b. Select the **Validate input data file** check box.
 - c. In the **Schema file for validation** box, enter the file path to the schema that you want to use to validate the test data source.
 - d. If the schema specifies a target namespace (using the targetNamespace attribute), enter the namespace in the **Schema target namespace for validation** box.
4. To validate the production data source that is used at run time, complete the following steps:
 - a. Click the **Production Data Source** tab.
 - b. Select the **Validate input data file** check box.

- c. To specify the schema that you want to use to validate the production data source that is used at run time, complete one of the following tasks:

| To | Do this |
|--|---|
| Use the same schema that is used during testing and include it in the package file | <p>i. Select the Include test validation schema in package for production use check box.</p> <p>ii. To define the location to which the engine writes the packaged schema file, complete the following steps:</p> <p>A. In the Write packaged schema file to box, enter the file path or symbolic name of the location where you want the engine to write the schema file during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with <code>DD:.</code> Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>B. If you use a symbolic name, when you run the engine, you must use the <code>PACKAGED_FILEMAP</code> engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.</p> <p>For more information about the <code>PACKAGED_FILEMAP</code> engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine writes the packaged schema to the specified location and then reads the schema from that location during production.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Tip: If you have already included a schema in the package file, but you later want to specify a different schema, you can use the <code>FILEMAP</code> engine switch to specify a different schema for use during production.</p><p>For more information about the <code>FILEMAP</code> engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p></div> <p>iii. If you want to use a different target namespace for production, enter it in the Schema target namespace for validation box. Otherwise, the namespace that you specified in the Schema target namespace for validation box on the Test Data Source tab automatically appears in the Schema target namespace for validation box on the Production Data Source tab.</p> |

| To | Do this |
|---|---|
| Use a different schema from the one that is used during testing | <p>i. To define the schema that you want to use to validate the production data source, complete the following steps:</p> <p>A. In the Schema file for validation box, enter the file path or symbolic name of the schema that you want to use to validate the production data source. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with <code>DD:</code>. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>B. If you use a symbolic name, when you run the engine, you must use the FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine reads the schema file that you specify from that location during production.</p> <p>ii. If the schema specifies a target namespace (using the <code>targetNamespace</code> attribute), enter the namespace in the Schema target namespace for validation box.</p> |

5. From the Menu bar select **Edit > Save**.

5.2.7 Transforming Data at Testing and Run Time

The data in a data source might not natively have the structure that is required by the Exstream Design and Production engine to create output. If you do not want to manually change the data structure before sending it to the engine, you have two options for automatically transforming the unstructured data before the engine reads it: connector objects and Extensible Stylesheet Language Transformations (XSLTs). You can use the two options together or separately. Keep in mind, though, that the ability to transform unstructured data is available only for input XML data files (for example, customer driver files or reference files) and schema model data files. You cannot transform Live data files, PDF form data files, or web service data files.

If you have licensed the Dynamic Data Access module, you can select a connector object to point to a routine that transforms incoming data before the data reaches the engine. The routine that is referenced by the connector object manipulates the data on a record-for-record basis or in a buffer (for tasks such as changing the record order).

For more information about creating connector objects, see *Configuring Connectors* in the Exstream Design and Production documentation.

XSLTs manipulate XML content when the content is moved between XML documents, XML schemas, or both. You can use an XSLT to transform unstructured data into the structure that the engine requires in order to produce an application. You are not required to pre-transform data before sending it to the engine. Instead, you can specify an XSLT that the engine uses to transform the data into the correct structure. By running the transform from the engine, and not externally, you can make changes to the transform in order to maintain the consistency of the data that is sent to the engine without updating a data file and without repackaging. (If you must change the structure of the data that is sent to the engine, you must update the corresponding data file and repackage the application.) Keep in mind, though, that if the engine transforms data, engine performance might decrease during testing and production.

You can optionally include the XSLT that you use for testing in the package file. This lets you use the same XSLT for testing and production, and also lets you keep it with the application so that you do not have to provide access to or distribute a separate file after packaging.

Caution: To transform XML data on the z/OS platform, you must first download and install the "XML Toolkit for z/OS" from the IBM website. Then, you must point your engine to run Job Control Language (JCL) to the installed Partitioned Data Set Extended (PDSE).

If you do not already have an XSLT, or if you want to create an XSLT for a specific application, you can create a sample XML data layout that includes all of the variables in an application, except system variables and function variables. The sample data layout provides you with the variables, in XML format, so that you can create a sample file for mapping or an XSLT. By using a design to drive the layout of your data files, you can optimize the data file layout for mapping, testing, and production. You can ensure that only the data that you require is being read by the engine and you can ensure that the data is in the correct order, both of which make processing more efficient.

For more information about creating a sample XML customer driver file, see “[Using a Variable Report to Create a Sample XML Data Layout](#)” on page 210.

To transform data at testing and run time:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Advanced** tab.
3. To assign a connector that points to a routine that transforms incoming data before it reaches the engine, complete the following steps as needed:

| To | Do this |
|-------------------------------------|--|
| Use the connector during testing | From the Test record transform box, select an existing connector object from the Library. |
| Use the connector during production | From the Production record transform box, select an existing connector object from the Library. |

4. To use an XSLT to transform data, complete the following steps:
 - a. Select the **Run XSLT engine on input** check box.
 - b. To use a variable to specify a dynamic XSL file, select a variable in the **XSL file location variable** box.
 - c. If you want to test output before going to production, enter the path to the XSL file that you want to use for testing in the **Test XSL file location** box.
 - d. To specify the XSLT that you want to use during production, complete one of the following sets of steps:

| To | Do this |
|--|--|
| Use the same XSLT that is used during testing and include it in the package file | <ol style="list-style-type: none">i. Select the Include test XSL file in package for production use check box.ii. To define the location to which the engine writes the packaged XSL file, complete the following steps<ol style="list-style-type: none">A. In the Write packaged test XSL file to box, enter the file path or symbolic name of the location where you want engine to write the XSLT file during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.<div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p>Tip: On a z/OS platform, it is a best practice to precede the symbolic name with DD:. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>The engine writes the packaged XSLT to the specified location and then reads it from that location for use in production.</p> <ol style="list-style-type: none">B. If you use a symbolic name, when you run the engine, you must use the PACKAGED_FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform. <p>For more information about the PACKAGED_FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine writes the packaged XSL file to the specified location and then reads the XSL file from that location during production.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p>Tip: If you have already included an XSL file in the package file, but you later want to specify a different XSL file, you can use the FILEMAP engine switch to specify a different XSL file for use during production.</p></div> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |

| To | Do this |
|---|---|
| Use a different XSLT from the one that is used during testing | <p>i. To define the XSL file that you want to use, complete the following steps:</p> <p>A. In the XSL file location variable box, enter the file path or symbolic name of the XSLT that you want to use during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.</p> <div style="border: 1px solid black; padding: 5px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with DD:. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>B. If you use a symbolic name, when you run the engine, you must use the FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine reads the XSL file that you specify from that location during production.</p> |

5. From the Menu bar, select **Edit > Save**.

When you produce Multi-Channel XML, keep in mind the embedded XSLT engine in Exstream supports XSLT version 1.0.

5.2.8 Using Attributes to Identify the XML Hierarchy

XML formatted data files use both tag attributes and tag names to define data. By default, Design Manager identifies the hierarchy using tag names. In some instances, however, the layout might have non-descriptive or generic tag names. To create a more descriptive hierarchy, you can substitute attributes for the tag name during mapping. A more descriptive hierarchy is especially important when you are producing output such as report files.

The following example compares the differences between how Design Manager identifies the hierarchy using the default names and how Design Manager identifies the hierarchy when it substitutes attributes.

XML hierarchy identification comparison

| Original Layout | Default Names | Attributes Substituted |
|--|---|---|
| <pre><statement period="08/08"> <customer> <category key="demographics"> <item key="name">Mary Smith</item> <item key="address">503 Yorkshire Drive</item> </category> <category key="account"> <item key="num">0012345</item> <item key="balance.opening">3500</item> <item key="balance.closing">3750</item> </category></pre> | <pre>statement customer category item</pre> | <pre>statement customer demographics name address account num balance.opening balance.closing</pre> |

During automapping, Design Manager uses the attributes to create variables. Because variable names cannot have spaces, make sure that the attributes you are substituting do not have spaces.

To use attributes to identify the XML hierarchy:

1. In Design Manager, from the Library, drag the XML formatted data file to the Property Panel.
2. Click the **Advanced** tab.
3. Select the **Replace tag name with attribute value** check box.
4. In the **Attribute name** box, enter an attribute value to replace the tag name.
5. From the **Edit** menu, select **Save**.

5.2.9 Defining the Structure of an XML Formatted Output Data File

If you are creating an auxiliary layout file, a report file, or a post-sort report file, you can use the data file properties to define the structure of the data the engine creates.

To define the structure of an XML formatted output data file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. From the **Encoding** list, select the encoding you want to include in the declaration tag. Select the option that most closely matches your native code page, or enter your own encoding. If you do not want to declare an encoding, leave this option blank.
3. In the **Indentation for each tag level** box, enter the number of spaces (from 0 to 10) to indent for each tag level in the XML output.
4. To suppress indentation and carriage returns or line feeds in XML output in order to reduce

the size of the file, select the **Lean XML** check box.

5. From the **Edit** menu, select **Save**.

5.2.10 Using a Variable Report to Create a Sample XML Data Layout

If you have licensed the XML/JSON Input module, you can use a variable report to create a sample XML data layout. The sample data layout includes the variables used in the design which have values that must come from a data file. For best results, keep in mind that the sample data layout is as complete as your design. If you want a sample XML data layout that includes all of the variables in the application, the design should be complete, or nearly complete, before you run the variable report. By using a design to drive the layout of your data files, you can optimize the data file layout for mapping, testing, and production. You can ensure that only the data you require is being read by the engine and you can ensure that the data is in the correct order, both of which makes processing more efficient.

The sample data layout is not meant to be an exact representation of a data file. Instead, it provides you with the variables, in XML format, so you can create a sample file for mapping or an Extensible Stylesheet Language Transformation (XSLT). You can use an XSLT to create a mapping, testing, or production data source. Before sending data to the engine, you can use an XSLT to transform data into the structure the engine requires in order to produce an application. You are not required to pre-transform data before sending it to the engine. Instead, you can specify an XSLT that the engine uses during production to transform the data into the correct structure.

For more information about using an XSLT during processing, see “[Transforming Data at Testing and Run Time](#)” on page 235.

To use a variable report to create a sample XML data layout:

1. In Design Manager, in the Library, right-click an application and select **Variable Report**.

The variable report opens in the Edit Panel.

2. To filter the report by approval status, design user, or effective dates, complete the following steps:

- a. Click **Object Selection Settings**.

The **Object Selection Version Settings** dialog box opens.

- b. Use the options on the dialog box to specify which variables are to be included in the report:

- i. From the **Version method** list, select **Version status**.

The **Includes selected status and any statuses above it** slider area appears.

- ii. Move the slider to one of the following options:
 - **Approved**—Includes all objects with a status of Approved or Archived. If you select **Approved**, you can choose to substitute quick fix versions by selecting the **Substitute Quick Fix versions** check box.
 - **Submitted for approval**—Includes all objects with a status of Submitted for Approval, Approved, and Archived
 - **Work in progress**—Includes all objects with a status of Work in Progress, Submitted for Approval, Approved, and Archived
 - **Rejected**—Includes all objects with a status of Rejected, Work in Progress, Submitted for Approval, Approved, and Archived
 - iii. If you want the report to include only objects from a specific user, and you selected **Submitted for Approval**, **Work in Progress**, or **Rejected** on the **Includes selected status and any statuses above it** slider, select the **Include versions from the following user** check box.
 - iv. Under the **Include versions from the following user** check box, click  .
The **Select Design User** dialog box opens.
 - v. From the list of design users, select a design user.
 - vi. Click **OK**.
The design user that you selected appears in the box under the **Include versions from the following user** check box.
 - vii. From the **Effective date** list, select how effectiveness dates affect which objects can be selected for packaging:
 - **As of Now**—Selects the latest valid version of the objects as of the current date (and time, if used)
 - **As of Date**—Selects the latest valid versions of the objects as of a specified date (and time, if used). Select the date and time from the adjacent **To** box.
 - **Date Range**—Selects the valid version of objects as of a range of dates (and times, if used). Select the date and time from the adjacent **From** and **To** boxes.
 - viii. Click **OK**.
The **Object Selection Version Settings** dialog box closes and the settings you selected appear in the **Version** and **Effective** boxes.
3. Right-click the Edit Panel and select **Export List**.
- The **Save As** dialog box opens.

4. In the **Save in** box, select the folder in which to save the sample data layout.
5. In the **File name** box, enter the name of the sample data layout.
6. From the **Save as type** list, select **Sample driver XML (*.xml)**.
7. Click **Save**.

5.2.11 Integrating XML Schemas with XML Formatted Data Files

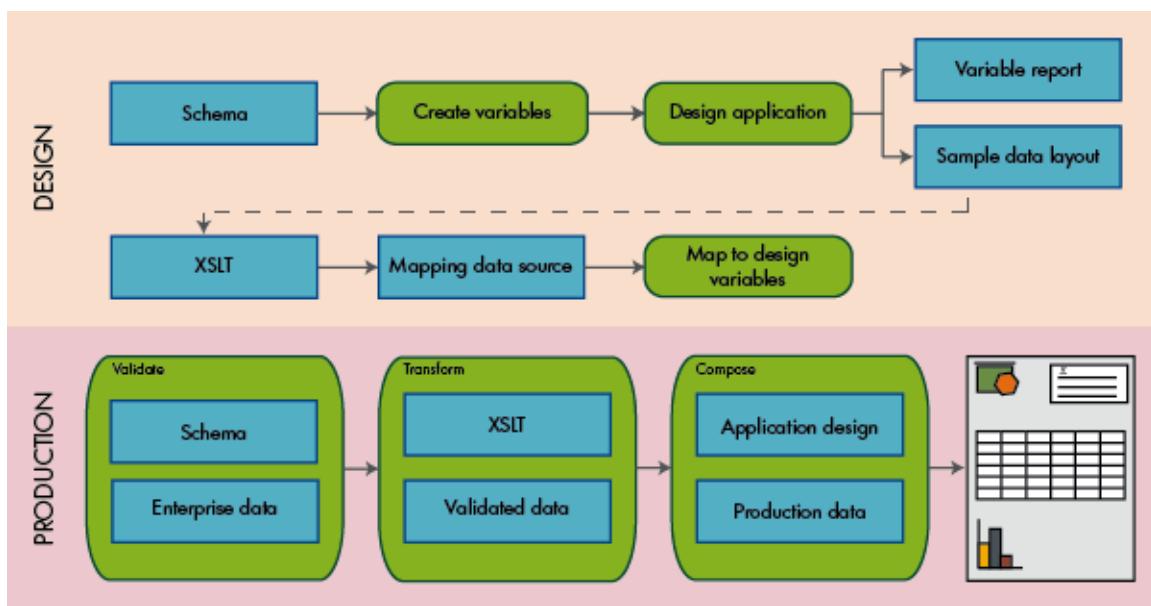
Note: Keep in mind that integrating an XML schema with an XML formatted data file is not the same as using a schema model data file and does not offer identical functionality.

For more information about using schema model data files in Exstream Design and Production, see “[Creating Schema Model Data Files](#)” on page 215.

In Exstream Design and Production, you can integrate XML schemas with XML formatted data files. Schemas define the structure of the building blocks that make up an XML file, and include information such as the data types and default values for the available elements and attributes. Because the XML data file defines the data, and the schemas that you integrate into the data file define only the structure of the data, integrating schemas with XML formatted data files allows you to better coordinate your design and data.

As shown in the following graphic, you can use schemas to automatically create variables and thus design without the sample data that is normally required for data mapping. You can use the design to create a transform that structures data into a mapping data source. You can then map the variables that are used in the design to the mapping data source. During production, you can also use schemas to validate unstructured enterprise data. The engine can then transform the validated data into the structure that is required for production.

Example of using an XML schema in a design and production process



If you have XML data that is prepared for mapping and you have not yet created variable objects, you can use a schema to create the variable objects automatically. When you create variable objects during auto-mapping, Design Manager reads the file or the schema and selects the variable object name, data type, and formatting features of a variable based on the data source. Design Manager maps the new or existing variable objects to the data automatically. If a schema defines data types that do not match data types in Exstream Design and Production, Design Manager substitutes a supported data type when it creates the variable objects. You can use schema-created variable objects alone or in combination with other variables.

The following steps represent a common design process that is completed when integrating an XML schema with an XML formatted data file:

| Step | Task | Description |
|------|--|---|
| 1 | Create variables automatically using the schema, and add metadata that identifies the variables as coming from the schema. | <p>When you are designing, mapping data and variables might not yet be available, which normally means that you must manually create variables in order to design. However, if you are using an XML schema, you are not required to have mapping data and you are not required to manually create variables. Instead, because schemas can define data types and values, you can use a schema to automatically create variables directly from the Data Dictionary, even if you do not have mapping data. When you create variables automatically, Design Manager reads the schema and selects the basic name, data type, and format features of a variable based on information in the schema. Additionally, you can add metadata to variables during automatic creation. For example, you might add a metadata object that identifies the schema used to create the variables so that you can filter the Variable Palette (by clicking ) to view only those variables later in the design process. The variables that you create using a schema are the same as the variables that you create manually or those that you create as you map a data file. You can use these schema-created variables alone or in combination in a design, just as you would other variables.</p> <p>For more information about using a schema to create variables, see "Creating Variables Automatically from XML or an XML Schema" on page 74.</p> |
| 2 | Design your application. While working in Designer, you can filter the Variable Palette to show only the variables with the metadata you added. | <p>For information about designing your application and inserting variables into a design, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> |
| 3 | Create a sample XML layout from a variable report. You might filter the report for "Work in Progress" variables to include only the newly created variables. | <p>After you have designed an application, you can create a sample XML data layout that includes the variables used in the design which have values that must come from a data file. The sample data layout provides you with the variables, in XML format, so you can create a sample file for mapping or an Extensible Stylesheet Language Transformation (XSLT). During design, you can use an XSLT to create a mapping data source. By using a design to drive the layout of your data files, you can optimize the data file layout for mapping, testing, and production. You can ensure that only the data you require is being read by the engine and you can ensure that the data is in the correct order, both of which make processing more efficient.</p> <p>For more information about creating a sample data file, see "Using a Variable Report to Create a Sample XML Data Layout" on page 210.</p> |
| 4 | Autemap the sample data file using the schema. | <p>If you used a schema to create variables instead of creating variables during automapping, you must still map the schema-created variables to a mapping data source before going into production using a data file. The variables you map to the data file must match the variables created using the schema and the variables used in the design.</p> <p>For more information about mapping a data file, see "Mapping an XML Formatted Data File" on page 189.</p> |

During production, you can use the schema to validate that the data conforms to the standard defined by the schema. You can then use an XSLT to transform the data into the structure that the design requires. The XSLT can transform the data during the engine run, so you are not required to transform the structure of the data prior to a production run. During production, if the schema validation fails, you receive error messages that describe the nature of the failure.

The following table represents common production tasks that are completed when integrating an XML schema with an XML formatted data file:

| Task | Description |
|--|--|
| Validate the data structure using the schema | <p>During production, you can use a schema to validate that the data conforms to the standard that is defined by the schema. Because you can transform the data into its final structure during processing, the data is not required to be structured before it is validated. The engine can validate unstructured data before it is transformed or, if you are not transforming unstructured enterprise data, the engine can validate structured data that is sent directly to the engine. You can use different schemas to validate data during testing and during production, and neither is required to match the schema that you used to map the data file. You can optionally include the production schema in the package file so that you do not have to provide access to or distribute a separate file after packaging.</p> <p>For more information about using a schema to validate data, see "Validating a Data Source at Testing and Run Time Using a Schema" on page 232.</p> |
| Use an XSLT to transform unstructured data into the structure that the engine requires | <p>Before sending data to the engine, you can use an XSLT to transform unstructured data into the structure that the engine requires in order to produce an application. You are not required to pre-transform data before sending it to the engine. Instead, you can specify an XSLT that the engine uses during production to transform the data into the correct structure. By separating the structure of the source data from the data mapping layout, you can make changes to your transform, as necessary, in order to maintain the consistency of the data sent to the engine. (If you must change the structure of the data sent to the engine, you must update the corresponding data file and repackage the application.) You can optionally include the XSLT in the package file so that you do not have to provide access to or distribute a separate file after packaging.</p> <p>For more information about transforming data, see "Transforming Data at Testing and Run Time" on page 235.</p> |

5.3 Creating Schema Model Data Files

Note: Keep in mind that using a schema model data file is not the same as integrating an XML schema with an XML formatted data file and does not offer identical functionality.

For more information about integrating an XML schema with an XML formatted data file in Exstream Design and Production, see ["Integrating XML Schemas with XML Formatted Data Files" on page 212](#).

To create a schema model data file, you must first create a data file object that defines the data file type and format. Then, you use the data file properties to define the general properties that apply to all of the data in the data file.

Schema model data files are able to identify the hierarchy of the data that is already defined in your data file, so that you do not have to use attributes or define the data structure in order to pre-define the hierarchy of a schema model data file in the same way that you would for other XML formatted data files.

To create a schema model data file object, you must complete the following tasks:

1. [“Considerations for Setting Up a Schema Model Data Source” below](#)
2. [“Creating a Schema Model Data File Object” on page 220](#)
3. [“Mapping a Schema Model Data File” on page 222](#)

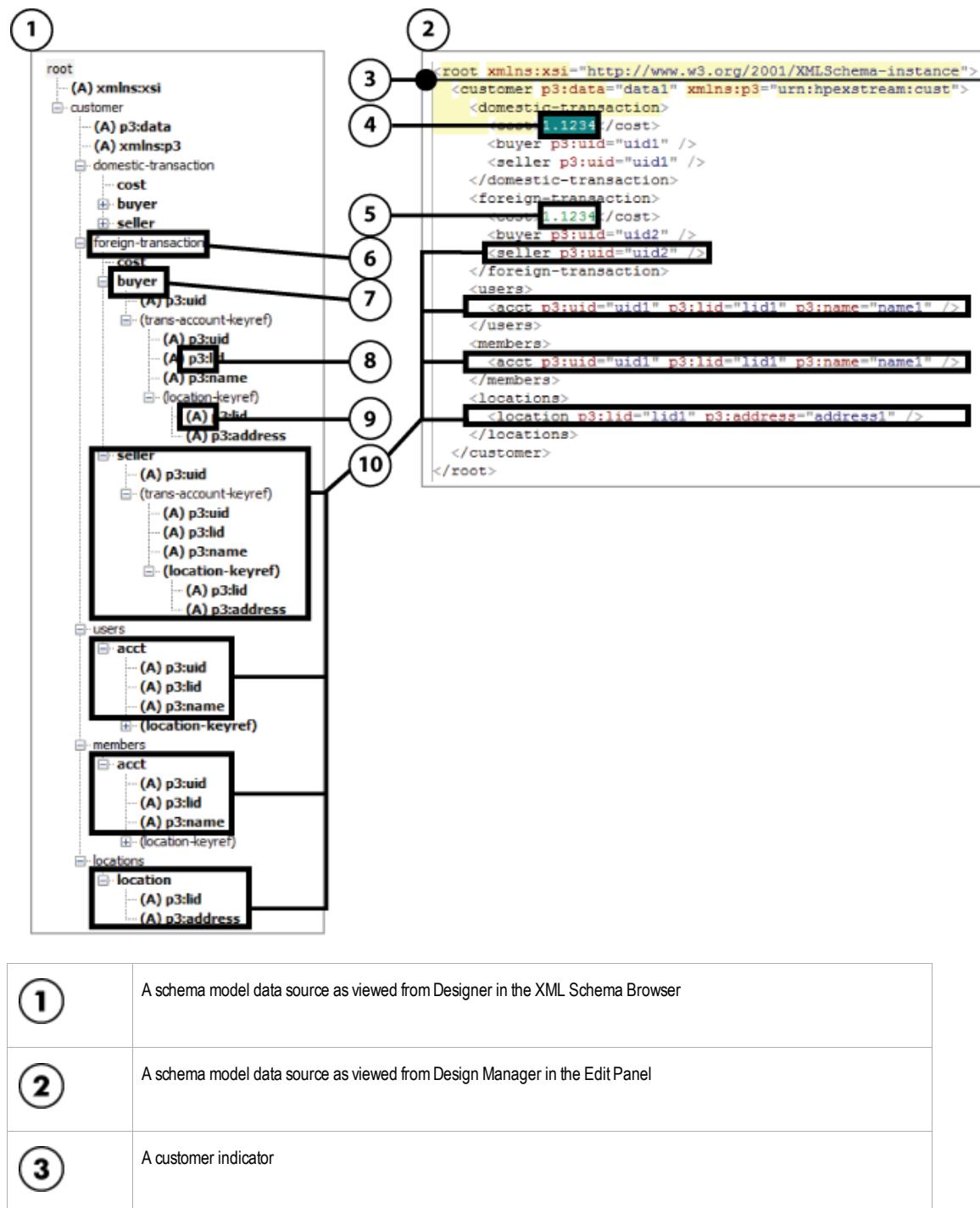
You can also complete the following optional tasks as needed:

- [“Changing Schema Model Data File Sources” on page 230](#)
- [“Validating a Data Source at Testing and Run Time Using a Schema” on page 232](#)
- [“Transforming Data at Testing and Run Time” on page 235](#)

5.3.1 Considerations for Setting Up a Schema Model Data Source

As with the other data file formats, you use a schema model data file object to define and point to existing data sources that you create and store outside of Exstream Design and Production. The structure of the schema model data source defines how you create and map a schema model data file in Exstream. When you interact with the data source in Exstream, you define the start of customers and identify the areas in which the engine can find data. As shown in the following graphic, the different parts of the schema data source structure work together to define the layout of data in Exstream.

An example of a schema model data source as viewed from Designer in the XML Schema Browser and from Design Manager in the Edit Panel



| | |
|------|---|
| (4) | A mapped value Notice that even though the value appears mapped in the Edit Panel, the same value does not appear mapped in the XML Schema Browser. You can view and edit mapped tags only from Design Manager in the Edit Panel. |
| (5) | An unmapped value |
| (6) | A non-mappable value |
| (7) | A mappable value Leaf nodes and attributes appear in bold in the XML Schema Browser to indicate that you can map them. |
| (8) | A namespace prefix If an attribute exists in a different namespaces than the node or if the <code>attributeFormDefault</code> attribute is set to <code>qualified</code> , then the namespace prefix appears in the XML Schema browser. Otherwise, the namespace prefix is not visible from Design Manager in the Edit panel or from Designer in the XML Schema Browser. |
| (9) | An attribute value Attributes in the XML Schema browser are always indicated by (A). |
| (10) | Key and Key References (also known as key and keyref in the schema model data source) Notice that if you view key and key reference data from Design Manager in the Edit Panel, you can view key and key references as they might appear in the data source. However, if you view the same key and key reference data from Designer in the XML Schema Browser, the key and key reference lookups are completed automatically and you can view all key or key reference information from any location where that data is referenced. If you want to view or map any values from the completed key reference lookup, you must view the schema model data source from Designer in the XML Schema Browser. |

Tip: In addition to using indicators to identify the different parts of the XML structure, Exstream also uses different colors in Design Manager in the Edit panel to indicate data file mappings and data behaviors.

To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select **View > What do colors mean**.

When you set up your schema model data source, keep in mind the following considerations:

| Data Source Consideration | Description |
|-----------------------------|---|
| Data structure requirements | <ul style="list-style-type: none">Schema model data sources that you use with Exstream Design and Production must be well-formed XML documents that comply with the W3C XML 1.0 specification. However, there is one limitation: you cannot have any text that comes after a child tag. The following sample compares those structures that are supported in Exstream Design and Production with a structure that is not supported in Exstream Design and Production. <p>Sample supported and unsupported XML structures</p> <div style="border: 1px solid black; padding: 10px;"><p>Supported</p><p>a) <a>lorem ipsum</p><p>b) <a>lorem ipsum<c>Dolor</c></p><p>c) <a>Dolor lorem ipsum</p><p>Unsupported</p><p><a>lorem ipsum Dolor</p></div> <ul style="list-style-type: none">Keep in mind that only one sample record is required for data mapping.Do not include blank tags that contain no tag values; instead, use a character such as an x to represent data areas.If you must ensure that a file is valid in addition to being well-formed, you can use either the <code>Driver.xsd</code> schema or a custom schema to structure an XML formatted data file. The <code>Driver.xsd</code> schema is available in your installation directory. Schemas define the structure of the building blocks that make up an XML file. They include information such as the available elements and attributes, the data types for the elements and attributes, and the default values for the elements and attributes. Data types are especially important in Exstream Design and Production, since you can use the data type information in the schema to create variables and to define the format of the data. In addition, you can use a schema to validate data coming from a data source and to create transforms that let you create XML data files with the exact structure that you require. <p>For more information about using schemas, see "Integrating XML Schemas with XML Formatted Data Files" on page 212.</p> |
| Encoding support | <ul style="list-style-type: none">The encoding of the schema model data source is identified only by the encoding string in the data source. Because of this source requirement, make sure that your XML is valid and well-formatted.If you use UTF-8 encoding for a schema model data file, keep in mind the following requirements:<ul style="list-style-type: none">You must be able to correctly view the characters using the active Windows code page. During packaging, Design Manager automatically includes this code page, and the engine uses it to map the UTF-8-encoded characters. If you use a code page other than Windows 1252, you must select the correct font and script in order to view the characters in the Edit Panel.If you are creating an SBCS schema model data file and the active code page is DBCS, Exstream Design and Production supports only characters with values 0 through 127. To use all the characters in a DBCS code page correctly, create a DBCS schema model data file.If you include the encoding attribute in the declaration tag, it must be <code>encoding="UTF-8"</code>.Exstream Design and Production does not create Byte Order Markers (BOMs) in output XML formatted data files. BOMs are not necessary, but you can include them in input XML formatted data files. |

| Data Source Consideration | Description |
|--|--|
| Transferring a data source between platforms | If you transfer a schema model data source between platforms (such as z/OS, Windows NT, and UNIX), view the data in the new platform to verify that the file transferred correctly. When you transfer a schema model data source, the line endings might change if you transfer in ASCII format. This change can adversely affect the way in which Exstream Design and Production parses the schema model data source, and can cause run-time errors if the ends of lines are incorrectly presented on the new platform. |
| Key and Key Reference Support | Schema model data files support key and key references in the data source. In Design Manager, in the Edit Panel, you can view key and key references exactly as they are written in the data. In Designer, in the XML Schema Browser, the key and key reference lookups are completed automatically and you can view all key or key reference information from any location where that data is referenced. If you want to view or map any values from the completed key reference lookup, you must view the schema model data source from Designer in the XML Schema Browser. |
| Data aggregation support | Schema model data files are processed during the data aggregation engine timing in order to gain broader access to customer data during processing. During the data aggregation engine timing, the engine has access to all of the customer data at one time. This engine timing allows the engine to read all of the customer data, including any XML node data, and to hold the data in memory until it reaches the next customer. However, you cannot use the data aggregation process to eliminate redundancy with schema model data files. You can only use the data aggregation process to eliminate redundancy in XML formatted data files that contain data sections. For more information about the data aggregation process, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation. |

5.3.2 Creating a Schema Model Data File Object

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **Type** list, select **Customer driver file**.
4. From the **Format** list, select **Schema model**.
5. Click **Finish**.

The **New Data File** dialog box closes and the schema model data file opens in the Property Panel for you to define.

6. On the **Basic** tab, complete the following steps to specify the schema source for the data file:
 - a. In the **Schema** box, click  and select the schema source file that you want to apply to the data file.
 - b. In the **Root element** list, select the root element that selects the portion of the

schema that you want to use. The options in the list can vary based on the root elements in the schema that you select. The root element that you select also determines which nodes are available when you map the data file.

Caution: If you change the root element, any data mapping that is applied to the schema model data file can be lost.

7. On the **Test Data Source** tab, complete one of the following sets of steps to specify the location of the XML data source to use for testing and, optionally, for mapping:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in test box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use in test box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

8. On the **Production Data Source** tab, complete one of the following sets of steps to specify the location of the XML data source to use during production:

| To | Do this |
|--------------------------------|--|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in production box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. |

Note: On a z/OS platform, it is a best practice to precede the symbolic name with **DD:.** Additionally, the remaining portion of the name must be eight or fewer characters.

For more information about the FILEMAP engine switch, see *Switch Reference* in the Exstream Design and Production documentation.

| To | Do this |
|--|---|
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">From the Type list, select Connector.In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

9. From the Menu bar, select **Edit > Save**.

5.3.3 Mapping a Schema Model Data File

You must manually map a schema model data file. Manual mapping includes tasks that you must complete for all data files, such as defining the start of a customer. You cannot automatically map a schema model data file.

Keep in mind that you are not required to map variables to all of the tags in a data file. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory that is required by the engine, map only those XML nodes and attributes that you want to use to populate variables.

This section discusses the following topics:

- “[Manually Mapping a Schema Model Data File](#)” below
- “[Viewing the Data Mapping Layout](#)” on page 271

Manually Mapping a Schema Model Data File

Schema model data files are unique among data files in that they can be mapped in Designer, as well as in Design Manager. You can simultaneously map customer data in schema model data files in Designer while you are creating customer content (such as mapping transaction data while creating a table of that data in the design). Or, as with other data files, you can manually map schema model data files in Design Manager to define rule- and logic-specific settings in the data.

This section discusses the following topics:

- “[Identifying Customers in Schema Model Data Files](#)” on the next page
- “[Mapping and Applying Text Variables from a Schema Model Data File While Designing Customer Content](#)” on the next page
- “[Manually Mapping a Variable to a Node Value or Attribute Value in a Schema Model Data File](#)” on page 227

Identifying Customers in Schema Model Data Files

For schema model data files, you must specify which XML node starts a new customer. You must identify at least one XML node as a customer in the data; otherwise, the engine will not process your application.

To identify a customer in a schema model data file:

1. In Design Manager, from the Library, drag the schema model data file to the Edit Panel.
2. In the data, select the unmapped customer node name or the value that represents the customer.
3. Right-click the highlighted data and select **Node > Properties**.

The **XML Node Mapping Properties** dialog box opens.

4. To indicate that the node starts a new customer, select the **Starts customer** check box.

NOTE: By default, the **Starts customer** check box is cleared. This setting indicates that the selected node does not start a customer.

5. Click **OK**.

The **XML Node Mapping Properties** dialog box closes and a solid black line appears above each instance of the node to indicate the start of a new customer. Each customer ends when the engine encounters the next customer node in the schema model data file.

6. From the Menu bar, select **Edit > Save**.

Mapping and Applying Text Variables from a Schema Model Data File While Designing Customer Content

In Designer, you can simultaneously map and apply variables from a schema model data file to text in your design as you create customer content in your application. Mapping schema model data files as you work in Designer helps to speed up the design process by letting you single-source the design and data mapping of customer data. For example, transaction statement tables are commonly driven by multiple variables that originate from customer driver files. By accessing the schema model data file that contains your customer data from Designer, you can simultaneously create the cell contents for a transaction statement table at the same time that you create and map the customer variables to the data file.

The XML Schema browser is a dockable palette that you can use to view and map schema model data in Designer. The XML Schema browser is best used to simultaneously map and apply new text variables to customer content through a click-and-drag or double-click interface. While you can also select and apply variables that are already mapped in a schema model data file to the text, it is a best practice to use the Variable palette to select and apply existing variables to the text in your design.

For more information about using the Variable palette, see *Designing Customer Communications* in the Exstream Design and Production documentation.

Keep in mind that mapping schema model data files from Designer is intended only for mapping and applying new text variables to customer content. If you want to map existing variables or create new variables for use with logic or formulaic calculations, it is a best practice to manually map the variables in the schema model data file from Design Manager.

For more information about manually mapping variables in a schema model data file from Design Manager, see “[Manually Mapping a Variable to a Node Value or Attribute Value in a Schema Model Data File](#)” on page 227.

To map and apply variables from a schema model data file while designing customer content:

1. In Designer, open the design into which you want to insert the customer content.
2. If the XML Schema browser is not visible, open the XML Schema browser by selecting **View > XML Schema browser** from the Menu bar.
3. If you want to filter the data files in the XML Schema browser, complete one or more of the following steps, depending on the filter criteria that you want to use:

| To | Do this |
|---|---|
| Filter schema model data files by the folder in which they are located in the Library | <p>a. Click  and select one of the following options:</p> <ul style="list-style-type: none">• Selected Folder—Show schema model data files from a specific folder in the Library.• Selected folder and Parents—Show schema model data files from a specific folder and each parent folder up to the root folder in the Library. <p>The Folders dialog box opens.</p> <p>b. Select a folder from the list and click OK.</p> <p>To show schema model data files from all folders in the Library, click  and select All Folders.</p> |
| Filter schema model data files by the application in which they are used | <p>a. Click  and select Selected Application.</p> <p>The Select Application dialog box opens.</p> <p>b. In the Look in box, select the folder where you want to look for the application.</p> <p>c. Select an application from the list and click OK.</p> <p>To show schema model data files used in all applications in the Library, click  and select All Applications.</p> |

4. From the **Select data file** list, select the schema model data file that you want to use.

The XML Schema browser is populated with the schema model data file that you select. Keep in mind that the file might require some time to load in Designer, depending on the size of the data file.

5. Filter or navigate the schema model data file to find the node that you want to map to the content. To do this, complete the following steps as needed:

| To | Do this |
|--|--|
| Show only nodes that match your criteria | <ol style="list-style-type: none">a. In the Filter box at the top of the browser, enter a node name. You can use an asterisk (*) as a wild-card character to represent unknown characters in a node name. To filter the content in the browser to show only the attributes, enter (A). To find a particular attribute, enter (A) : followed by the attribute name, or enter just the attribute name.b. Click  or press ENTER. <p>The browser shows only the parts of the tree that contain the nodes that match the criteria that you entered.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: To determine whether a particular node in the filtered view is the node that you want to map, you might need to see its position in the fully expanded tree. To do this, click the node in the filtered view to highlight it and then click  to clear the filter. The node will be highlighted in the expanded view.</p></div> |
| Expand an individual node | Click  or double-click the node. |
| Expand all nodes at a specific level in the data | <p>Note: Double-clicking to expand a node is available only if you are expanding a node with children or attributes. If you double-click a node with no children and no attributes, then the node is mapped and the variable is added to the text.</p> |
| Collapse all nodes at a specific level in the data | <ol style="list-style-type: none">a. Select a node at the level of data that you want to collapse.b. Right-click the node and select Expand all at this level. <p>All of the sibling or cousin nodes in the data file that are at the same data level as the node that you selected are expanded.</p> |
| | <ol style="list-style-type: none">a. Select a node at the level of data that you want to collapse.b. Right-click the node and select Collapse all at this level. <p>All of the sibling or cousin nodes in the data file that are at the same data level as the node that you selected are collapsed.</p> |

6. To map and apply variables from a schema model data file to text to the design, complete the following steps as needed:

| To | Do this |
|--|---|
| Simultaneously map a node or attribute value and insert a new variable in the XML Schema browser by clicking and dragging the node | <ol style="list-style-type: none">In the XML Schema browser, click and drag the node that you want to map.Drop the node into the design at the text location where you want to insert the text variable that you create. <p>Note: If the variable name is invalid, the Rename variable dialog box opens, and you are prompted to rename the variable. A variable name can be invalid if the variable name contains an invalid character, if the variable name contains more than 255 characters, or if the variable name is a duplicate of an existing variable name.</p> <p>The text variable is created, mapped to the node that you selected, and inserted into the text.</p> |
| Simultaneously map a node or attribute value and insert a new variable in the XML Schema browser by double-clicking the node | <ol style="list-style-type: none">In the design, place the cursor at the location where you want to insert the text variable.In the XML Schema browser, double-click the node that you want to map. <p>Note: If the variable name is invalid, the Rename variable dialog box opens, and you are prompted to rename the variable. A variable name can be invalid if the variable name contains an invalid character, if the variable name contains more than 255 characters, or if the variable name is a duplicate of an existing variable name.</p> <p>The text variable is created, mapped to the node that you selected, and inserted into the text.</p> |

7. From the Menu bar, select **File > Save**.

Keep in mind the following considerations when you create and map variables from Designer:

- By default, the new variable is automatically named using the following format: <Parent Node>_<Node name>. For example, if the path to the node is root\Customer\Checking\AccountNumber, then the variable would be named Checking_AccountNumber. Variable names can have a maximum of 255 characters. If a variable name exceeds the character limit, then the variable name is truncated.
- The XML Schema Browser automatically places variables in the same folder as the design page. Because of this variable placement, if your organization uses user permissions, you must have a minimum **Functional access level** of **Create** on the folder in order to create and add new variables using the XML Schema Browser. Alternately, if you have **Create** permissions on a different folder, you can map the variable to that folder from Design Manager using the Edit Panel and then use the Variable Palette in Designer to insert the mapped variables into the design page.

- You can view the node path of a mapped variable by hovering the pointer over the variable to display a tooltip that contains the name of the variable and the path to the node that is mapped to the variable. Keep in mind that while the node path does contain the information that Exstream needs in order to locate the node in the data, the path is not a valid XPath location.

Tip: If you cannot see tooltips in Designer, make sure that you have selected the **Show popup tips** check box on the **Designer** tab of the **Designer Options** dialog box. To open the **Designer Options** dialog box, from the Menu bar, select **Tools > Options**.

- All mappings that you create from Designer use default node properties. You cannot change the node properties of a mapping from Designer. If you want to modify the node properties you must open and edit the schema model data file in Design Manager.

For more information about manually mapping variables in a schema model data file from Design Manager, see “[Manually Mapping a Variable to a Node Value or Attribute Value in a Schema Model Data File](#)” below.

- All mappings that you create from Designer use default data area properties. You cannot change the data area properties of a mapping from Designer. If you want to modify the data area properties you must open and edit the schema model data file in Design Manager.

For information about specifying how the engine uses a data area during processing, see “[Specifying How the Engine Uses the Data Area During Processing](#)” on page 314.

For information about specifying the input format of the data, see “[Specifying the Input Format of Data in a Data Area](#)” on page 154.

- If you need to review the data mapping of the file after you have finished creating the customer content, you can open the data file in the Edit Panel in Design Manager and review the file mapping.

For more information about viewing the data mapping layout see “[Viewing the Data Mapping Layout](#)” on page 271.

Manually Mapping a Variable to a Node Value or Attribute Value in a Schema Model Data File

Note: For SBCS schema model data files, in order to view the file accurately in the Edit Panel, you must select **Native** from the **Character set** list in the data file properties. If you select a different option from the **Character set** list, you can receive unexpected results.

1. In Design Manager, from the Library, drag the schema model data file to the Edit Panel.
2. In the data, select any unmapped node value or attribute value.
3. Right-click the highlighted data and select **Node > Properties**.

The **XML Node Mapping Properties** dialog box opens.

4. To map the node, do one of the following:

| To | Do this |
|---|---|
| Map an existing variable to a node value | <ol style="list-style-type: none">a. Click the Data mapped to node value box. A prompt opens, asking whether you want to use an existing variable.b. Click Yes. The Select Variable dialog box opens.c. Select the variable from the list.d. Click OK. The Data Area Properties dialog box opens. |
| Map a new variable to a node value | <ol style="list-style-type: none">a. Click the Data mapped to node value box. A prompt opens, asking whether you want to use an existing variable.b. Click No. The New Variable dialog box opens.c. In the Name box, enter a name. In the Description box, enter a description (optional).d. From the Type list, select the type of data that the variable represents.e. If the variable can contain more than one value, select the Array check box.f. In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples.g. Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Map an existing variable to the attribute value | <ol style="list-style-type: none">a. Enter the attribute value in the Value box. By default, the attribute value in the node populates the box.b. To specify that the exact value must always be found within the node for this mapping to apply, select the The attribute/value pair is required for the mapping check box.c. Click the Data mapped to value box. A prompt opens, asking whether you want to use an existing variable.d. Click Yes. The Select Variable dialog box opens.e. Select the variable from the list.f. Click OK. The Data Area Properties dialog box opens. |

| To | Do this |
|---|--|
| Map a new variable to the attribute value | <ol style="list-style-type: none">a. Enter the attribute value in the Value box. By default, the attribute value in the node populates the box.b. To specify that the exact value must always be found within the node in order for this mapping to apply, select the The attribute/value pair is required for the mapping check box.c. Click the Data mapped to value box. A prompt opens, asking whether you want to use an existing variable.d. Click No. The New Variable dialog box opens.e. In the Name box, enter a name. In the Description box, enter a description (optional).f. From the Type list, select the type of data that the variable represents.g. If the variable can contain more than one value, select the Array check box.h. In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer. Exstream Design and Production does not support Unicode for design samples.i. Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Skip the selected node during processing | <p>Select the Ignore this node check box. If you select this option, all of the other options on the dialog box are disabled. Skip to step 7.</p> |

5. In the **Data Area Properties** dialog box, verify or update the properties for the data area.

For information about specifying how the engine uses a data area during processing, see [“Specifying How the Engine Uses the Data Area During Processing” on page 314](#).

For information about specifying the input format of the data, see [“Specifying the Input Format of Data in a Data Area” on page 154](#).

6. Click **OK**.

The **Data Area Properties** dialog box closes.

7. To remove white space at the beginning or end of the node value, select the **Suppress whitespace** check box.

8. Click **OK**.

The **XML Node Mapping Properties** dialog box closes and Design Manager applies a color to the node value to indicate that you have mapped the name to a variable. The color of the data mapping can vary depending on the settings that you apply. To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select **View > What do colors mean**. In addition, you can also view information about an individual mapped data area (such as the variable name) when you hover the pointer over a mapped data area.

For more information about data area properties, see “[Using a Data Area to Specify How the Engine Reads and Writes Data](#)” on page 151.

Viewing the Data Mapping Layout

You can view details about the data mapping layout of a data file. For example, you can review the properties of all of the variables that are mapped to the data file.

To view the data mapping layout:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click and select **View > View Layouts**.
A list of the tags or elements in the layout opens in the Property Panel.
3. Click a specific tag or element, and if it is present in the mapped data file, it is highlighted in the Edit Panel.

5.3.4 Changing Schema Model Data File Sources

When you select the data source for a schema model data file, you define the location of the data you use for mapping, testing, and production.

A test data source is typically a subset of the production data that you use to test and troubleshoot an application before putting it into production. To ensure that your tests are as accurate as possible, the test data source and the production data source should be similar. During a test, the engine uses the data maps that you created using the mapping data source, along with the test data source, to read and write customer data. You can then use the results to make sure that the final application design is accurate.

A production data source is the complete set of data required to produce the output that you send to customers. During production, the engine uses the data maps that you created using the mapping data source, along with the production data source, to read and write customer data.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

To change the schema model data file sources:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. In the **Schema** box, click  and select the schema source file that you want to apply to the data file.
4. In the **Root element** list, select the root element that selects the portion of the schema that you want to use. The options vary based on the root elements in the schema. The root element also determines which nodes are available when you map.

5. Click the **Advanced** tab.
6. In the **Rule** box, enter a rule to include or exclude the file during an engine run.
7. On the **Test Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use for testing and, optionally, mapping:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in test box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use in test box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

8. On the **Production Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in production box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with DD:. Additionally, the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

9. From the **Edit** menu, select **Save**.

When you save the data file, Exstream checks the file to make sure that all of the node definitions from the original data source still exist in the data file. If any of the node definitions have changed, when you save, Exstream issues a warning. If you accept the warning, any variable mappings that are applied to the missing nodes are lost. If you cancel the warning, repeat step 2 through step 4 and select a different source that contains the missing node structure.

5.3.5 Validating a Data Source at Testing and Run Time Using a Schema

During testing and production, the engine reads data from the data source. If you are defining an input XML data file (for example, customer driver files or reference files) or a schema model data file, you can use a schema to ensure that the data source is valid and conforms to the standard that is defined by the schema. The engine can validate unstructured data before it is transformed or, if you are not transforming unstructured enterprise data, the engine can validate structured data that is sent directly to the engine. Because you can transform the data into its final structure during processing, the data is not required to be structured before it is validated.

For more information about transforming data, see “[Transforming Data at Testing and Run Time](#)” on page 235.

You can use different schemas to validate data during testing and during production, and neither schema is required to match the schema that is used to map the data file. Keep in mind that if you validate data, the engine performance might decrease during testing and production.

You can optionally include the schema that you used for testing in the package file in order to use the same schema for production, and to keep the schema with the application so that you do not have to provide access to or distribute a separate file after packaging.

Caution: To validate XML data on the z/OS platform, you must first download and install the “XML Toolkit for z/OS” from the IBM website. Then, you must point your engine to run Job Control Language (JCL) to the installed Partitioned Data Set Extended (PDSE).

If the engine finds an error, only the first instance of the first deviation from the schema is included in the engine error message file. This restriction helps to prevent the engine error message file from becoming too large. Also, keep in mind that you might receive errors if the declared encoding does not match the actual encoding of the schema, even if the data is valid.

Note: The task in this section assumes that you are using a schema model data file or that you are using an XML data file and have selected **Schema** from the **Data mapping method** list on the **Basic** tab. You must use one of these schema-compatible files in order to validate your data using a schema at testing and run time.

To validate the data source at testing and run time using a schema:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To validate the test data source, complete the following steps:
 - a. Click the **Test Data Source** tab.
 - b. Select the **Validate input data file** check box.
 - c. In the **Schema file for validation** box, enter the file path to the schema that you want to use to validate the test data source.
 - d. If the schema specifies a target namespace (using the targetNamespace attribute), enter the namespace in the **Schema target namespace for validation** box.
4. To validate the production data source that is used at run time, complete the following steps:
 - a. Click the **Production Data Source** tab.
 - b. Select the **Validate input data file** check box.

- c. To specify the schema that you want to use to validate the production data source that is used at run time, complete one of the following tasks:

| To | Do this |
|--|---|
| Use the same schema that is used during testing and include it in the package file | <ol style="list-style-type: none">i. Select the Include test validation schema in package for production use check box.ii. To define the location to which the engine writes the packaged schema file, complete the following steps:<ol style="list-style-type: none">A. In the Write packaged schema file to box, enter the file path or symbolic name of the location where you want the engine to write the schema file during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.<div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with <code>DD:.</code> Additionally, the remaining portion of the name must be eight or fewer characters.</p></div>B. If you use a symbolic name, when you run the engine, you must use the <code>PACKAGED_FILEMAP</code> engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.<p>For more information about the <code>PACKAGED_FILEMAP</code> engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p><p>The engine writes the packaged schema to the specified location and then reads the schema from that location during production.</p><div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Tip: If you have already included a schema in the package file, but you later want to specify a different schema, you can use the <code>FILEMAP</code> engine switch to specify a different schema for use during production.</p><p>For more information about the <code>FILEMAP</code> engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p></div>iii. If you want to use a different target namespace for production, enter it in the Schema target namespace for validation box. Otherwise, the namespace that you specified in the Schema target namespace for validation box on the Test Data Source tab automatically appears in the Schema target namespace for validation box on the Production Data Source tab. |

| To | Do this |
|---|---|
| Use a different schema from the one that is used during testing | <p>i. To define the schema that you want to use to validate the production data source, complete the following steps:</p> <p>A. In the Schema file for validation box, enter the file path or symbolic name of the schema that you want to use to validate the production data source. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with <code>DD:</code>. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>B. If you use a symbolic name, when you run the engine, you must use the FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine reads the schema file that you specify from that location during production.</p> <p>ii. If the schema specifies a target namespace (using the <code>targetNamespace</code> attribute), enter the namespace in the Schema target namespace for validation box.</p> |

5. From the Menu bar select **Edit > Save**.

5.3.6 Transforming Data at Testing and Run Time

The data in a data source might not natively have the structure that is required by the Exstream Design and Production engine to create output. If you do not want to manually change the data structure before sending it to the engine, you have two options for automatically transforming the unstructured data before the engine reads it: connector objects and Extensible Stylesheet Language Transformations (XSLTs). You can use the two options together or separately. Keep in mind, though, that the ability to transform unstructured data is available only for input XML data files (for example, customer driver files or reference files) and schema model data files. You cannot transform Live data files, PDF form data files, or web service data files.

If you have licensed the Dynamic Data Access module, you can select a connector object to point to a routine that transforms incoming data before the data reaches the engine. The routine that is referenced by the connector object manipulates the data on a record-for-record basis or in a buffer (for tasks such as changing the record order).

For more information about creating connector objects, see *Configuring Connectors* in the Exstream Design and Production documentation.

XSLTs manipulate XML content when the content is moved between XML documents, XML schemas, or both. You can use an XSLT to transform unstructured data into the structure that the engine requires in order to produce an application. You are not required to pre-transform data before sending it to the engine. Instead, you can specify an XSLT that the engine uses to transform the data into the correct structure. By running the transform from the engine, and not externally, you can make changes to the transform in order to maintain the consistency of the data that is sent to the engine without updating a data file and without repackaging. (If you must change the structure of the data that is sent to the engine, you must update the corresponding data file and repackage the application.) Keep in mind, though, that if the engine transforms data, engine performance might decrease during testing and production.

You can optionally include the XSLT that you use for testing in the package file. This lets you use the same XSLT for testing and production, and also lets you keep it with the application so that you do not have to provide access to or distribute a separate file after packaging.

Caution: To transform XML data on the z/OS platform, you must first download and install the "XML Toolkit for z/OS" from the IBM website. Then, you must point your engine to run Job Control Language (JCL) to the installed Partitioned Data Set Extended (PDSE).

If you do not already have an XSLT, or if you want to create an XSLT for a specific application, you can create a sample XML data layout that includes all of the variables in an application, except system variables and function variables. The sample data layout provides you with the variables, in XML format, so that you can create a sample file for mapping or an XSLT. By using a design to drive the layout of your data files, you can optimize the data file layout for mapping, testing, and production. You can ensure that only the data that you require is being read by the engine and you can ensure that the data is in the correct order, both of which make processing more efficient.

For more information about creating a sample XML customer driver file, see “[Using a Variable Report to Create a Sample XML Data Layout](#)” on page 210.

To transform data at testing and run time:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Advanced** tab.
3. To assign a connector that points to a routine that transforms incoming data before it reaches the engine, complete the following steps as needed:

| To | Do this |
|-------------------------------------|--|
| Use the connector during testing | From the Test record transform box, select an existing connector object from the Library. |
| Use the connector during production | From the Production record transform box, select an existing connector object from the Library. |

4. To use an XSLT to transform data, complete the following steps:
 - a. Select the **Run XSLT engine on input** check box.
 - b. To use a variable to specify a dynamic XSL file, select a variable in the **XSL file location variable** box.
 - c. If you want to test output before going to production, enter the path to the XSL file that you want to use for testing in the **Test XSL file location** box.
 - d. To specify the XSLT that you want to use during production, complete one of the following sets of steps:

| To | Do this |
|--|--|
| Use the same XSLT that is used during testing and include it in the package file | <ol style="list-style-type: none">i. Select the Include test XSL file in package for production use check box.ii. To define the location to which the engine writes the packaged XSL file, complete the following steps<ol style="list-style-type: none">A. In the Write packaged test XSL file to box, enter the file path or symbolic name of the location where you want engine to write the XSLT file during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.<div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p>Tip: On a z/OS platform, it is a best practice to precede the symbolic name with DD:. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>The engine writes the packaged XSLT to the specified location and then reads it from that location for use in production.</p> <ol style="list-style-type: none">B. If you use a symbolic name, when you run the engine, you must use the PACKAGED_FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform. <p>For more information about the PACKAGED_FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine writes the packaged XSL file to the specified location and then reads the XSL file from that location during production.</p> <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p>Tip: If you have already included an XSL file in the package file, but you later want to specify a different XSL file, you can use the FILEMAP engine switch to specify a different XSL file for use during production.</p></div> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |

| To | Do this |
|---|---|
| Use a different XSLT from the one that is used during testing | <p>i. To define the XSL file that you want to use, complete the following steps:</p> <p>A. In the XSL file location variable box, enter the file path or symbolic name of the XSLT that you want to use during production. Symbolic file names let you run the same production file on different platforms using a control file. When you run the engine, you map these symbolic file names to physical names in a control file using an engine switch.</p> <div style="border: 1px solid black; padding: 5px;"><p>Note: On a z/OS platform, it is a best practice to precede the symbolic name with DD:. Additionally, the remaining portion of the name must be eight or fewer characters.</p></div> <p>B. If you use a symbolic name, when you run the engine, you must use the FILEMAP engine switch in a control file to map the symbolic file name to a physical path and file name that is valid for the production platform.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> <p>The engine reads the XSL file that you specify from that location during production.</p> |

5. From the Menu bar, select **Edit > Save**.

When you produce Multi-Channel XML, keep in mind the embedded XSLT engine in Exstream supports XSLT version 1.0.

5.3.7 Using XML Nodes to Drive an Application

This section walks you through the settings that you must apply if you want to use the XML nodes in the data structure of your schema model data file to drive how content is included in the customer output. When you use an XML node to drive content inclusion, you link objects or content in a design to XML nodes in the schema model data source. Then, each time that node is encountered in the customer data, the node affects how the connected object, content, or logic is included in the output. You have multiple controls over how the nodes are connected to content and what actions are applied when the engine encounters the node in the data.

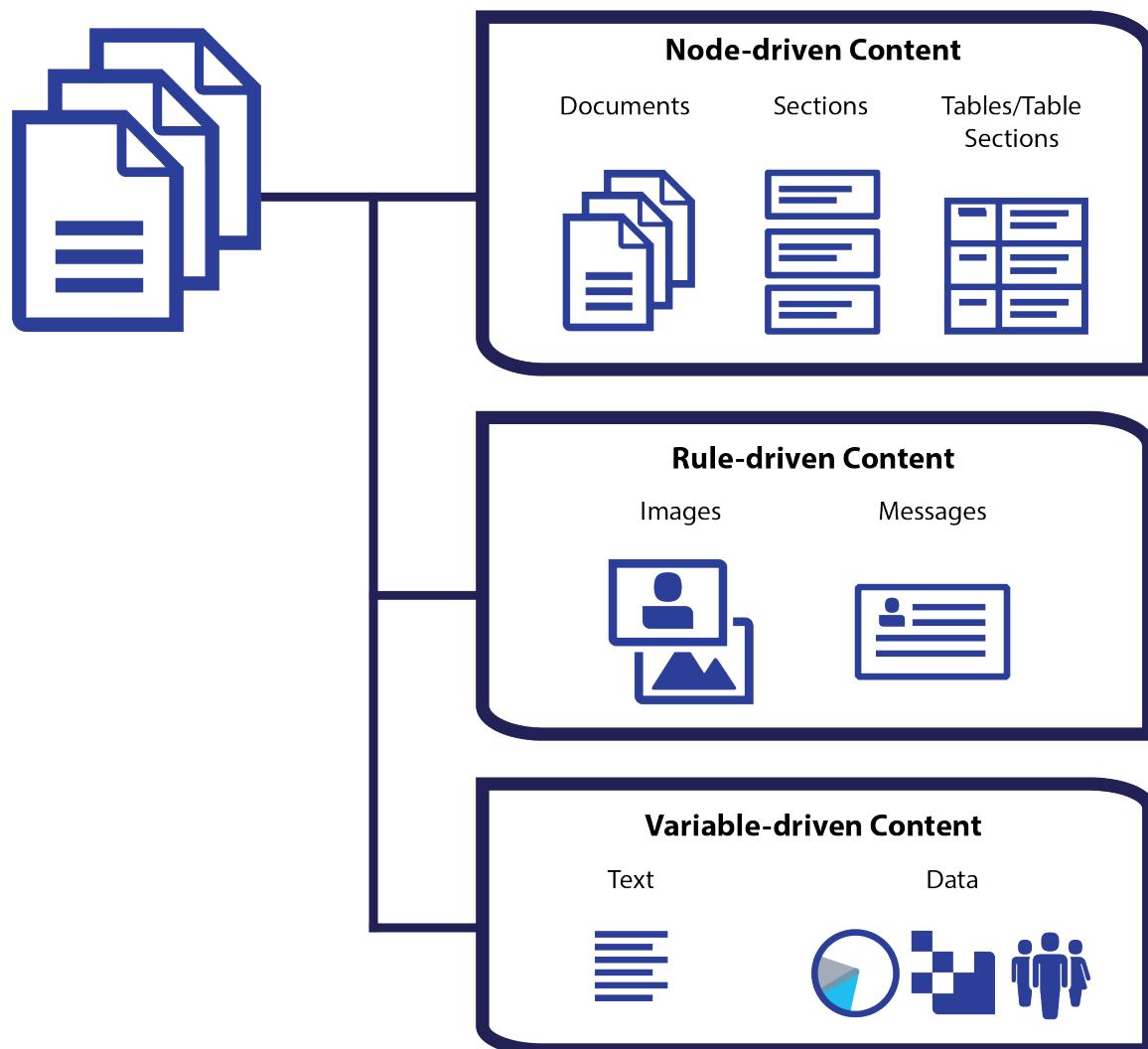
The following table provides a high-level overview of the tasks (along with links to the sections that discuss them) that you can follow to incorporate XML nodes into your design:

| Task | Description | Related Information |
|---|--|---|
| In the documentation, determine whether including objects based on XML nodes is the optimal content inclusion method for the results that you want to achieve. | XML nodes are one of multiple options available in Exstream to control the inclusion of content in your application. Review the documentation to quickly determine whether you should enable this functionality for specific content in your application or not. | "When Should I Use XML Nodes to Drive My Application?" on the next page |
| In the documentation, review limitations and notes about special use cases. | Before you implement XML node-driven content in your application, review the related information so that you can avoid any pitfalls. | <ul style="list-style-type: none"> • "Special Considerations for Using XML Nodes to Drive an Application" on page 242 • "Timing Considerations for XML Nodes" on page 243 |
| In your document objects, determine which documents you want to include based on node data. | <p>If you want to use XML nodes to drive document inclusion you must select which node in the data file triggers the inclusion of the document.</p> <p>In addition, you can also specify the inclusion behavior that the engine should apply if the data contains consecutive nodes.</p> | "Sending a Document to Customers Based on an XML Node" on page 244 |
| In your section objects, determine which section and paragraph objects should be included based on node data. | If you want to use XML nodes to drive section object inclusion, you must select which node in the data file triggers the inclusion of the section object. | "Sending a Section to Specific Customers Using An XML Node" on page 245 |
| In your design, in any table objects, specify how node data should be used to drive the creation of tables. | If you want to use XML nodes to drive the creation of table content, you must select which nodes in the data file drive the content of the table. | "Using XML Node Data To Create Tables" on page 245 |
| If you need more complex access to your node data, you can use the EvaluateXPath built-in function in your logic to process more complex logic in your data. | The EvaluateXPath built-in function lets you evaluate the data within a schema model data file by using XPath expressions in order to gain broader access to data within a schema model data file than you might be able to through typical mapping or filters. | Using Logic to Drive an Application in the Exstream Design and Production documentation |

When Should I Use XML Nodes to Drive My Application?

Keep in mind that this section outlines only those tasks that are needed to use node data to drive the inclusion of content in an application. There are alternate ways to drive object inclusion, including rules, formulas, functions, locales, and more. If using a node to drive object inclusion does not meet your needs, try an alternate method of content inclusion.

Example of an application with XML node-driven content



| Inclusion Method | Description | Related Information |
|------------------|--|--|
| XML Nodes | <p>You can include documents and sections based on whether an XML node exists in the customer data. You can also use XML nodes to drive the creation of tables.</p> <p>XML nodes are available in data files that meet all of the following criteria:</p> <ul style="list-style-type: none">• The Data File Type must be Customer Driver File.• The Data Format must be Schema Model. | <p>For more information about setting up a schema model data file, see "Creating Schema Model Data Files" on page 215.</p> |

| Inclusion Method | Description | Related Information |
|--|--|--|
| Data Sections (also called Named sections) | <p>Schema model data files do not support data sections, so you cannot have data sections in a data file that contains XML nodes.</p> <p>Data Sections are available in data files that meet all of the following criteria:</p> <ul style="list-style-type: none"> The Data File Type must be either Customer Driver File or Reference. The Data File Format must be either XML, Columnar, or Delimited. <p>However, since you can use multiple customer driver files on a single application, keep in mind that it is a best practice not to mix driving an application with both XML node-based data files and data section-based data files.</p> | For more information about data sections, see "Section Data in Exstream Design and Production" on page 4 . |
| Rules | <p>Rules use a mixture of data and logic to determine whether specific content within a design is included or excluded for specific customers.</p> <p>Define and apply rules as needed in your application.</p> | For more information about creating rules and rule logic, see Using Logic to Drive an Application in the Exstream Design and Production documentation. |
| Variables | <p>Variables either directly reference customer data as it appears in the data file (such as customer names and transaction data) or variables are customized to contain logic that calculates specific values during customer processing (such as page numbers or customer counts).</p> <p>Map and define variables as needed in your application.</p> | <p>For more information about mapping variables in a schema model data file, see "Mapping a Schema Model Data File" on page 222.</p> <p>For more information about customizing variables, see "Setting Up Variables" on page 59.</p> |

Special Considerations for Using XML Nodes to Drive an Application

When you design an application that uses XML nodes to drive content, keep in mind the following considerations:

| Application feature | Description |
|--|--|
| DXF output support for XML nodes | XML nodes are not supported in DXF output. If you include XML node settings in an application that you deliver to DXF output, these settings are ignored. |
| Empower support for XML nodes | You can use logic and variables that are built using XML nodes in Empower. However, Empower does not support using XML nodes to drive the inclusion of documents or section objects. Also, because Empower supports only simple tables and basic automated tables, Empower does not support using XML nodes to drive the creation of advanced tables. |
| Interactive document support for XML nodes | <p>You can use XML node-driven content in Interactive documents, and end users can edit XML node-driven content in LiveEditor. Keep in mind, however, that end users cannot add XML node-driven objects to a Live document.</p> <p>For more information about the end user experience with XML nodes in Interactive documents, see "How XML Node-Driven Content Affects Editing in Live Documents" on the next page.</p> |

How XML Node-Driven Content Affects Editing in Live Documents

From LiveEditor, end users cannot add XML node-driven objects to a Live document; however, end users can edit objects that are driven by XML nodes. It is important to understand the scope of where end user edits and changes are applied for XML node-driven objects and content.

When you set up XML nodes and the application, keep the following considerations in mind:

| Consideration | Description |
|--|--|
| Deleting XML node-driven objects from LiveEditor | If an end user deletes an object that is driven by XML nodes, the underlying data is not lost. However, the object will no longer appear in LiveEditor. |
| Using functions with XML nodes | If a function is fired within an object that is driven by an XML node, then that function affects only other objects that are driven by the same XML node. For example, if a function is run to update text inside an XML node-driven document, then only text in other documents that are driven by the same XML node is affected. |
| Using rules with XML nodes | Rules are run in the context of the XML node that you reference in the rule logic. Specifically, rules use the variable values within the scope of the XML node that you reference. The scope of the XML node can include the children of the XML node, key references, or the nearest relative of the XML node. |
| Using variables with XML nodes | The default number of elements in an array variable is zero. Therefore, a growing array variable will be empty until elements are assigned to it. If the array variable remains empty, it will not contain any editable content, and it will not drive automated content, such as table rows or paragraphs. |
| Using paragraph objects with XML nodes | For paragraph objects, if end users make edits to an interactive area, those edits are also applied to any other instance of the same paragraph object (regardless of whether that paragraph object is included in a section object that is driven by XML nodes), because the paragraph object is a single design object that is used in multiple locations. However, if you want to allow end users to make changes to a paragraph object that are unique to other instances of the same object, then you must use the following settings on the paragraph object: <ol style="list-style-type: none">1. You must include the paragraph object in an XML node-driven section object.2. In the schema model data file, you must map a variable as a child or descendant of the same XML node that drives the inclusion of the section object.3. In the paragraph object, include a reference to the variable that you added to the data file. End users can then edit the variable, and their changes to that variable would affect only that single instance of the paragraph object in the XML node-driven section object. |
| Using XML node-driven documents | Because the underlying data determines the order in which XML node-driven documents appear in a Live document, XML node-driven documents cannot be moved using the Outline Viewer panel in LiveEditor. For information about allowing end users to reorder documents in the Outline Viewer in LiveEditor, see <i>Designing for LiveEditor</i> in the Exstream Design and Production documentation. |

Timing Considerations for XML Nodes

When you are planning the engine timing of schema model data and XML nodes, keep in mind the following considerations:

| Timing consideration | Description |
|---|--|
| Engine timing for schema model data files and XML nodes is different than engine timing for general data and data sections. | <p>If you are working with schema model data files and XML node-driven content, it is important to review the engine process specific to XML node processing.</p> <p>XML node processing in the engine can differ from other applications. When the engine processes a schema model data file with XML nodes, the engine has access to all of the customer data at one time, allowing the engine to read all of the customer data, and to hold the data in memory until it reaches the next customer.</p> <p>By reviewing the engine process that is specific to XML node processing, you can make sure that your application is designed in a way that optimizes the results that you want in your output.</p> <p>For more information about engine timing for XML nodes, see <i>Preparing Applications for Production</i> in the Exstream Design and Production documentation.</p> |
| Make sure that objects and content are composed when they are needed based on engine timing. | <p>The term "compose time" refers to the time during the engine run at which objects are placed on a page. By default, all objects have a compose time of When page created; that is, they are placed on the page when the page is created. However, you can change the compose time of objects to accommodate other factors that affect the object, such as rules that are executed on the object later during the engine processing that determine whether the object is included in the output.</p> <p>For more information about Controlling When Objects Are Placed on a Page, see <i>Designing Customer Communications</i> in the Exstream Design and Production documentation.</p> |
| Make sure that variables that reference XML nodes use the appropriate compute time. | <p>For variables in schema model data files, it is a best practice to select a Compute time of As needed. This option optimizes the way in which the engine accesses variable data in schema model data files, because for schema model data files, all of a customer's data is made available at the beginning of the processing of that customer. This data availability means that variables in schema model data files are best computed as they are encountered in the design.</p> <p>For more information about setting up compute time for variables, see "Specifying the Variable Values" on page 65.</p> |

Sending a Document to Customers Based on an XML Node

If you use schema model data files, you can use XML nodes in the data structure to control when a document is included for a customer. Using an XML node to drive the inclusion of a document in customer output is useful when documents are similar in design but contain different types of data. For example, suppose that you design a document that you want to send only if a customer has a savings account. You can use the SAVINGS node of the data file to control the inclusion of the document. The document is included only if the SAVINGS node exists in the customer's data structure.

For more information about XML nodes and schema model data files, see *Using Data to Drive an Application* in the Exstream Design and Production documentation.

By default, if the engine encounters subsequent occurrences of the same XML node that drives document inclusion in the customer data, then the engine will always start a new document. For example, suppose that customers can have multiple SAVINGS nodes, each representing a unique account. Each of the customer savings accounts are delivered in a separate document.

To send a document to customers based on an XML node in the data structure:

1. In Design Manager, from the Library, drag the document to the Property Panel.
2. Click the **Targeting** tab.
3. From the **Method** drop-down list, select **XML node**.
4. In the **Location** box, click **</>** and select the node that you want to drive document inclusion.

The path to the node that you select appears in the **Location** box. Keep in mind that while the node path does contain the information that Exstream needs in order to locate the node in the data, the path is not a valid XPath location.

5. From the Menu bar, select **Edit > Save**.

Sending a Section to Specific Customers Using An XML Node

You can tie a section object to an XML node in the structure of the customer data in order to include a section object in customer output based on the presence of that XML node in the customer data. Each time that the engine encounters the XML node that you specify, a copy of the section object is included in the customer output.

To send a section object to specific customers based on an XML node in the customer data:

1. In Design Manager, from the Library, drag the section object to the Property Panel.
2. Click the **Targeting** tab.
3. From the **Inclusion Method** drop-down list, select **XML Node**.
4. In the **Location** box, click **</>** and select the node that you want to use to drive the inclusion of the section object in the customer output.

The path to the node that you select appears in the **Location** box. Keep in mind that while the node path does contain the information that Exstream needs in order to locate the node in the data, the path is not a valid XPath location.

5. From the Menu bar, select **Edit > Save**.

Using XML Node Data To Create Tables

You can use XML nodes to drive the creation of repetitive data in tables. Then, you can use table sections in order to present the XML node data in a categorized way. When using XML nodes to drive the creation of tables, keep in mind the following behaviors:

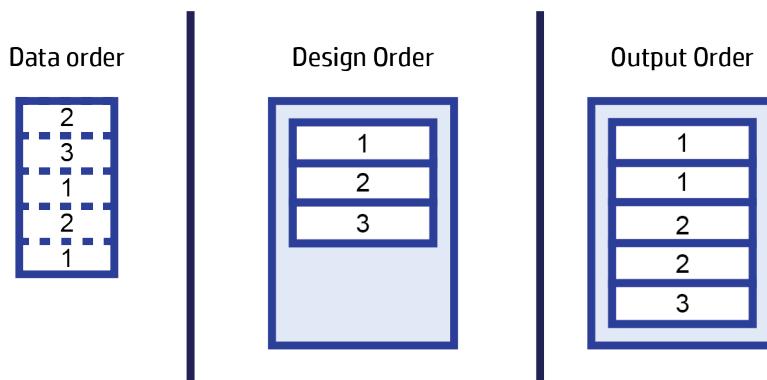
- While tables remain in the order in which they are designed on the page, table content that is driven by XML nodes is populated in the engine in the order that the data is processed by the engine. This data processing order makes it important for you to plan how you arrange and

flow content on the page compared to when content is processed by the engine.

For more information about engine processing with XML nodes, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

- The composed table sections are ultimately ordered based on the order of the table sections in the design, not based on the order of the XML node data. As illustrated in the following graphic, when you are using XML node-based data, the order that the XML nodes occur in the data are ignored, and the order of the table sections in your design control the order of the content in the output. All rows that are used to create each table must be related by sets or by groups to an XML node in the data. When creating complex tables, keep this principle in mind because the table might need to be restructured or have additional rules to compensate for the data order.

Example of the XML node order in the data file compared to the design and output



Tip: As you develop and test XML node-driven tables, you can use colors in the header and table rows to help you see how the data affects the table structure in the output.

- You must indicate the XML node on the first row of the table's section (whether the row type is a header, an automated row, or a footer).
- If you use XML nodes to drive the creation of tables that can split and flow, the table will split, flow, and re-paginate as tables normally do. However, the content of the cells is fixed from the time of the engine run.
- Headers and footers do not appear for customers without related data. Therefore, a table cannot appear with a header and footer but no content.
- The size of a table that is driven by XML nodes depends on the data in the customer driver file.
- Tables that are embedded within table cells cannot use XML node-driven rows.

For more information about XML nodes and schema model data files, see *Using Data to Drive an Application* in the Exstream Design and Production documentation.

Creating a Table with Sections that are Driven by XML Nodes

When you design tables with content that is driven by XML nodes, you must first set up the tables so that they contain the basic structure of the final table, including table sections. Next, you use table sections to define the areas of the table that are driven by XML nodes. Finally, you define the section data settings as needed for each table section.

For information about using table sections to organize table content, see *Designing Customer Communications* in the Exstream Design and Production documentation.

To create a table with table sections that are driven using XML nodes, you must use one of the following table types:

- Automated table with sections
- Automated table with levels
- User table

To create a table that is driven by XML nodes:

1. In Designer, select the table to which you want to apply XML nodes.

2. Click .

The **Table Properties** dialog box opens.

3. Click the **Table** tab.

4. Select the **Enable data sections/XML nodes** check box.

5. Click **OK**.

The **Table Properties** dialog box closes.

6. Select the first row in the table section.

7. Right-click the selected row and select **Row properties**.

The **Row Properties** dialog box opens.

8. Click the **Automated Row Properties** tab.

9. Use the options in the **Table section (set of rows)** area to define the properties of the table section.

10. From the **When to include section** drop-down list, select **XML Node**.

11. In the **Location** box, click **</>** and select the XML node that you want to use to drive the inclusion of the table section in the table.

The path to the XML node that you select appears in the **Location** box. Keep in mind that while the XML node path does contain the information that Exstream needs in order to locate the XML node in the data, the path is not a valid XPath location.

12. If you want to attach the section to the last section created by the engine (for example, to create a summary section that follows one or more transaction sections), select the **Add to previous table section (to combine data sections/XML nodes for flow)** check box.
This option is not available for headers.
13. Click **OK**.

The **Row Properties** dialog box closes.

EvaluateXPath

The EvaluateXPath function can be used only with schema model data files.

For more information about schema model data files, see *Using Data to Drive an Application* in the Exstream Design and Production documentation.

The EvaluateXPath function lets you evaluate the data within a schema model data file by using XPath expressions in order to gain broader access to the data that is within a schema model data file than you might be able to through typical mapping or filters.

For best results when using the EvaluateXPath function, you should be familiar with writing XPath expressions.

Syntax

```
EvaluateXpath(Array, "DataFileName", "DefaultNamespace", "XPathExpression",
[AdditionalNamespaces])
```

EvaluateXPath arguments

| Argument | Data type | Use | Description |
|------------------|-----------|----------|--|
| Array | Any | Required | The name of the array variable that you want to use to contain the returned data. |
| DataFileName | String | Required | <p>The name of the schema model data file that you want to evaluate.</p> <p>You can enter the value as a variable or as a string. You must enter string values in double quotation marks. Text that is entered without double quotation marks is interpreted as the name of a variable.</p> |
| DefaultNamespace | String | Required | <p>The namespace URI of the default namespace that is used in the schema model data file.</p> <p>For example:</p> <p><code>www.hp.com/go/exstream</code></p> <p>You can enter the value as a variable (either scalar or array) or as a string. You must enter string values in double quotation marks. Text that is entered without double quotation marks is interpreted as the name of a variable.</p> <p>If the data file content that you are evaluating does not belong to a namespace, you can use an empty string (" ") to represent the namespace URI.</p> |

EvaluateXPath arguments, continued

| Argument | Data type | Use | Description |
|----------------------|-----------|----------|---|
| XPathExpression | String | Required | <p>The XPath expression that you want to evaluate.</p> <p>As you define your XPath expression, keep in mind the following considerations:</p> <ul style="list-style-type: none">• All XPath expressions must be evaluated in the context of the root node.• If you want to limit the scope of the Xpath expression results to a specific customer, you must use a predicate to identify the customer node in the XPath expression. Otherwise, the function will evaluate the XPath expression against nodes from the entire data file.• If your data file uses multiple namespaces, you must specify the namespace prefix that is associated with each XPath location in the XPath expression. Otherwise, Exstream assumes that any location in the XPath expression that does not use a namespace prefix exists within the default namespace.• If there is a prefix in the XPath expression that does not belong to the namespace array, the function will fail.• Exstream does not validate the accuracy of your XPath code.• You can enter the value as a variable (either scalar or array) or as a string. You must enter string values in double quotation marks. Text that is entered without double quotation marks is interpreted as the name of a variable. |
| AdditionalNamespaces | String | Optional | <p>An array variable that specifies any additional namespace URIs and namespace prefixes that are used by the schema model data file.</p> <p>Within the array variable, the syntax of the namespace URIs and prefixes must appear in the following format:</p> <p><code>prefix:namespace URI</code></p> <p>For example:</p> <p><code>esx:www.hp.com/go/exstream</code></p> |

Returns

The EvaluateXPath function returns a count of the elements that are returned in the array that you specified in the Array argument for the function. The return value is in the integer data type.

In the array variable, array elements that were pulled from the schema model data file that you specified in the function remain in their original data formats. Array values in the array variable start at an index of 1.

Examples

There are many different ways that you can use the EvaluateXPath function. The following table includes different examples that illustrate the versatility of the EvaluateXPath function:

| Example | Description |
|----------------|--|
| Data Filtering | <p>Suppose that the data source for your schema model data file is arranged as follows:</p> <pre><BankStatements> <Customer Account="4172130"> <Name>Thomas Reagan</Name> <Transactions>30</Transactions> <Bank1 name="Cust_B1"> <Savings> <Debit date="080114" currency="USD">198.00</Debit> <Credit date="080414" currency="USD">398.48</Credit> <Bank1>Cust_B2</Bank1> </Savings> <Checking> <Debit date="080714" currency="USD">107.57</Debit> <Credit date="080814" currency="USD">387.16</Credit> <Credit date="080814" currency="USD">278.90</Credit> <Bank1>Check_B1</Bank1> </Checking> </Bank1> </Customer> </BankStatements></pre> <p>If you wanted to look up all the Bank1 credit transactions that were completed within the savings accounts for each customer in the schema model data file, the syntax for the EvaluateXPath built-in function might look similar to the following example:</p> <pre>EvaluateXPath(ResultsArrayVariable, "MyDataFile", "", "/BankStatements/Customer/Bank1[" & SYS_CustomerInRun & "]//Savings/Credit")</pre> <p>Notice that for this example, the XPathExpression argument uses the 'SYS_CustomerInRun' system variable in order to return results based on the customer that the engine is processing during the engine run when the function is encountered.</p> |

| Example | Description |
|---------------------------------------|--|
| Locate data outside the customer tags | <p>Suppose that the data source for your schema model data file contains data that is outside of the scope of any defined customer tags, but you need to access that data for use in your application. The data source for your schema model data file might be arranged as follows:</p> <pre data-bbox="540 397 980 635"><BankStatements xmlns="www.hp.com" xmlns:ex="www.hp.com/go/exstream"> <ex:Init> <ex:RunID>12345</ex:RunID> </ex:Init> <ex:Customer Account="4172130"> ... </ex:Customer> </BankStatements></pre> <p>If you needed to access the data in the <code><ex:RunID></code> tags, traditional mapping would not allow you to access that data because data mapping is customer-centric and the <code><ex:RunID></code> information exists outside of the customer tags. You will need to set up the <code>EvaluateXPath</code> function in order to access data that is outside of the customer data structure.</p> <p>Because the content also uses additional namespaces, you will also need to include a namespace array variable (<code>NamespaceArray</code>) that defines the following additional namespace and prefix from the data source:</p> <pre data-bbox="540 889 833 910">ex:www.hp.com/go/exstream</pre> <p>The syntax for the <code>EvaluateXPath</code> built-in function might look similar to the following example:</p> <pre data-bbox="540 952 1192 1005">EvaluateXPath(ResultsArray, "MyDataFile", "www.hp.com", "/BankStatements/ex:Init/ex:RunID", NamespaceArray)</pre> <p>Notice that for this scenario, the function does not include the '<code>SYS_CustomerInRun</code>' variable. Without the '<code>SYS_CustomerInRun</code>' variable, the function returns the same result set regardless of when or where the function is included during the engine run.</p> |

| Example | Description |
|---|---|
| Complete manual key/keyref lookups without having any key/keyref definitions in the data file | <p>Suppose that the data source for your schema model data file is arranged as follows:</p> <pre><root> <Policy> <Number>123456</Number> <PolicyEffDate>Jan 1 2015</PolicyEffDate> </Policy> <Policy> <Number>222222</Number> <PolicyEffDate>Aug 1 2012</PolicyEffDate> </Policy> <Insured> <ID>1</ID> <PolicyNum>222222</PolicyNum> </Insured> </root></pre> <p>Also suppose that you want to look up the effective date (<code><PolicyEffDate></code>) of a customer policy based on the policy number of a specific insured customer (<code><PolicyNum></code>). Based on the order of the data, the effective date and the insured customer's policy number are kept in different locations in the XML file and do not have a keyref defined for Exstream to perform the lookup automatically.</p> <p>If you want to set up the <code>EvaluateXPath</code> function to complete a manual key/keyref lookup, the function syntax for this scenario might look similar to the following example:</p> <pre>EvaluateXPath(ResultsArray, "MyDataFile", "", "/root/Policy/PolicyEffDate[../Number = /root/Insured[" & SYS_CustomerInRun & "]/PolicyNum"]")</pre> <p>Notice that for this example, the <code>XPathExpression</code> argument uses the 'SYS_CustomerInRun' system variable in order to return results based on the customer that the engine is processing during the engine run when the function is encountered.</p> |

Chapter 6: Setting Up JSON Formatted Data Files

In addition to the other data file formats, Exstream Design and Production also supports data file formats that use JSON to define the structure of the data. The JSON functionality of Exstream Design and Production lets you leverage existing infrastructure using a web service.

Setting up a JSON formatted data file, no matter the type or format, is similar to setting up a data file in other formats. This chapter discusses the following topics:

- “[Module Requirements for JSON Data Source Formats](#)” below
- “[Creating JSON Formatted Data Files](#)” below

6.1 Module Requirements for JSON Data Source Formats

To use the JSON formatted data source formats that are available in Exstream, you must license the modules that are related to your specific business requirements. The following table describes the data source formats that use JSON to structure data and the modules that are required for each data source format.

JSON formatted data file format module requirements

| Data file format | Required module | Related information |
|----------------------------|------------------------|---|
| JSON data file (as input) | XML/JSON Input | “Creating JSON Formatted Data Files” below |
| Web Service (RESTful) | Web Services Interface | “Setting Up Data Files to Communicate with a Web Service” on page 272 |
| JSON data file (as output) | XML/JSON (Data) Output | “Setting Up JSON Formatted Data Files” above |

For more information about Exstream Design and Production modules, see *Getting Started* in the Exstream Design and Production documentation.

6.2 Creating JSON Formatted Data Files

Creating a JSON formatted data file, no matter the type, is similar to creating a data file in other formats. First, you create a data file object that defines the data file type and format. Then, you

use the data file properties to define the general properties that apply to all of the data in the data file.

To create a JSON formatted data file, you must complete the following tasks:

1. [“Considerations for Setting Up a JSON Formatted Data Source” below](#)
2. [“Creating a Data File Object” on the next page](#)
3. [“Defining the Data Mapping Method” on page 256](#)
4. [“Selecting the Data File Sources” on page 257](#)
5. [“Mapping a JSON Formatted Data File” on page 260](#)

You can also complete the following optional task as needed:

- [“Defining the Structure of a JSON Formatted Output Data File” on page 271](#)

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

6.2.1 Considerations for Setting Up a JSON Formatted Data Source

You can use a JSON formatted data file to define and point to existing data sources that are created and stored outside of Exstream Design and Production. The structure defines how you create and map a JSON formatted data file. You use the structure to define the start of customers and sections and to identify the areas in which the engine can find data.

Tip: In addition to using indicators to identify the different parts of the JSON structure, Exstream also uses different colors in the Edit panel to indicate data file mappings and data behaviors.

To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select **View > What do colors mean**.

To help you set up your JSON formatted data source, keep in mind the following considerations:

| Data Source Consideration | Description |
|-----------------------------|---|
| Data structure requirements | <p>JSON formatted data files used with Exstream Design and Production must be well-formed JSON documents that comply with the JSON Data Interchange Standard.</p> <p>The hierarchy of a JSON formatted data file and the time at which the engine reads data affect each other. The engine reads or writes a JSON data file in the order it encounters the data. By default, it ignores element hierarchy and does not maintain relationships between the levels in the data. You can structure your data in such a way, or select how and when the engine reads data, in order to create the output you require.</p> |

| Data Source Consideration | Description |
|--|--|
| Encoding support | <p>Byte Order Markers (BOMs) are not valid in JSON and should not be included in JSON formatted data files.</p> <p>Exstream Design and Production has several encoding options that you can use with JSON formatted data files. UTF-8 encoding is also available as a character set in JSON formatted data files.</p> <p>For more information about selecting an encoding or a character set, see "Defining the Records in a Customer Driver File" on page 25.</p> <p>If you select the UTF-8 encoding for a JSON formatted data file, you must meet the following requirements:</p> <ul style="list-style-type: none">• You must be able to correctly view the characters using the active Windows code page. During packaging, Design Manager automatically includes this code page, and the engine uses it to map the UTF-8-encoded characters. If you use a code page other than Windows 1252, you must select the correct font and script in order to view the characters in the Edit Panel.• If you are creating an SBCS JSON formatted data file and the active code page is DBCS, the Exstream engine supports characters with values 0 through 127. To use all the characters in a DBCS code page without using an escape character, create a DBCS JSON formatted data file. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><p>Note: The \u escape character is supported for JSON formatted data files. If possible, the engine remaps the Unicode character value to the code page. If the character cannot not be mapped, the engine ignores the character value and issues a warning.</p></div> |
| Transferring a data source between platforms | <p>If you transfer a JSON formatted data source between platforms (such as z/OS, Windows NT, and UNIX), view the data in the new platform to verify that the file transferred correctly. When you transfer a JSON formatted data source, the line endings might change if you transfer in ASCII format. This change can adversely affect the way Exstream Design and Production parses the JSON formatted data source and can cause run-time errors if the ends of lines are incorrectly presented on the new platform.</p> |

6.2.2 Creating a Data File Object

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.
The **New Data File** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. Select the type and format of the data file you want to create.
4. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

For more information about which data file types are available for each format, see ["Data File Type and Format Availability" on page 14](#).

6.2.3 Defining the Data Mapping Method

One step in creating an XML or a JSON formatted data file is selecting how the data file will be mapped. The data mapping method determines how the variables are created as part of the data mapping process and the amount of customization that is available when you map.

For more information about mapping an XML or JSON formatted data file, see the following topics:

- “[Mapping an XML Formatted Data File](#)” on page 189
- “[Mapping a JSON Formatted Data File](#)” on page 260

To define the data mapping method:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To specify the method for data mapping, you must complete one of the following sets of steps:

| To | Do this |
|--|---|
| Automatically map the data file using a predefined schema or a schema contained in a Web Services Description Language (WSDL) file | <p>From the Data mapping method list, select Schema.</p> <p>Note: This option is available for XML data files, input Live data files, Web Service (RESTful) data files, and Web Service (SOAP) data files only. You cannot use a schema to map JSON data files, output Live data files or PDF form data files.</p> |
| Manually or automatically map the data file in the Edit Panel | <p>From the Data mapping method list, select Sample file (manual).</p> |
| Dynamically map the data file using a specified list of variable names that match the tag names | <ol style="list-style-type: none">a. From the Data mapping method list, select Sample file (simple). If you select this option, you cannot customize the data area format. Instead, the engine automatically uses the default settings. The engine also ignores data sections.b. In the Top level tag box, enter the top level tag. This tag is the start of the data file.c. In the Customer tag box, enter the tag used to indicate a new customer.d. In the Variables for simple mapping box, enter the variables used to map the XML data file. <p>Note: This option is not available for JSON data files.</p> |

| To | Do this |
|---|---|
| Dynamically map all of the tags in a data file using variable names that match the tag name | <ol style="list-style-type: none">From the Data mapping method list, select Sample file (dynamic). If you select this option, you cannot customize the data area format. Instead, the engine automatically uses the default settings. The engine also ignores data sections.In the Top level tag box, enter the top level tag. This tag is the start of the data file.In the Customer tag box, enter the tag used to indicate a new customer. <p>Note: This option is not available for JSON data files.</p> |

4. If you selected either **Sample file (simple)** or **Sample file (dynamic)** from the **Data mapping** list and are mapping a data file that contains at least one array, complete one of the following sets of steps, depending on how the array is defined:

| If the array is defined in this way | Do this |
|---|--|
| A group tag that matches the array variable name encloses the data elements. All of the data elements use the same tag. | <ol style="list-style-type: none">From the Array variable name handling list, select Use variable name for group.In the Array element tag box, enter the tag name for each data element in the array variable. |
| A group tag that matches the array variable name followed by a suffix encloses the data elements. The tag for each of the data elements matches the name of the array variable. | <ol style="list-style-type: none">From the Array variable name handling list, select Use variable name for element.In the Array group tag suffix box, enter the suffix that follows the variable name in the tag that introduces the group of data elements. For example, if you enter <code>_Array</code> and the name of the variable is <code>Debit</code>, the tag will be <code><Debit_Array></code>. |
| A single tag encloses all of the data elements for multiple array variables. The tag for each of the data elements matches the name of the array variable. | <ol style="list-style-type: none">From the Array variable name handling list, select Use for element, single group.In the Array group tag box, enter the name of the tag that encloses all of the data elements for all of the array variables. |

5. From the **Edit** menu, select **Save**.

6.2.4 Selecting the Data File Sources

When you select the data source for an XML or a JSON formatted data file, you define the location of the data you use for mapping, testing, and production.

A mapping data source is typically the simplest of the data sources you use. It contains sample data that helps you create data maps that identify the location of data and identify the start of a new customer or section. If you are mapping an XML or an input Live data file (for example,

customer driver files or reference files), you can use a schema as the mapping source. This option is not available for JSON data files.

A test data source is typically a subset of the production data that you use to test and troubleshoot an application before putting it into production. To ensure that your tests are as accurate as possible, the test data source and the production data source should be similar. During a test, the engine uses the data maps that you created using the mapping data source, along with the test data source, to read and write customer data. You can then use the results to make sure that the final application design is accurate.

A production data source is the complete set of data required to produce the output you send to customers. During production, the engine uses the data maps that you created using the mapping data source, along with the production data source, to read and write customer data.

For more information about mapping a data file, see [“Data File Mapping” on page 129](#).

To select the data file sources:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. To specify the location of the data source to use during mapping, complete one of the following sets of steps:

| To | Do this |
|---|--|
| Use a sample data file | <ol style="list-style-type: none">a. From the Data mapping method list, select Sample file (manual).b. In the Data mapping source box, enter the file path to the external file used to provide data. |
| Use a schema or a schema contained in a Web Services Description Language (WSDL) file | <ol style="list-style-type: none">a. From the Data mapping method list, select Schema.b. In the Data mapping source box, select a schema that defines the layout of the data or a WSDL file that contains a schema that defines the layout of the data.c. In the Root element list, select the root element that selects the portion of the schema that you want to use. The options vary based on the root elements in the schema. The root element also determines which variables are available when you map. |

Note: This option is available for XML data files, input Live data files, XML formatted Web Service (RESTful) data files, and XML formatted Web Service (SOAP) data files only. You cannot use a schema to map output data files, JSON data files, Live data files or PDF form data files.

4. Click the **Advanced** tab.
5. If data for a customer can change during processing (if you are using a connector to connect to the data source, for example) and you want to update the information in the data source, complete the following steps:

- a. Select the **File can be updated** check box.
- b. In the **Update output file name when running in test mode** box, enter the path to the output file.
- c. In the **Update output file name when running in production mode** box, enter the path to the output file.

Note: Use the Trigger function to control the timing of the update operation.

For more information about the Trigger function, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

6. If you are using XML data files, and the files you use include namespaces, make sure that the **Enable XML namespace support** check box is selected.
7. In the **Rule** box, enter a rule to include or exclude the file during an engine run.
8. On the **Test Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use for testing and, optionally, mapping:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">a. From the Type list, select File.b. In the File to use in test box, enter the file path to the external file used to provide data. |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">a. From the Type list, select Connector.b. In the File to use in test box, enter the file path to the external file used to provide data.c. In the Connector box, select an existing connector object from the Library. |

Note: The **Connector** option is available only if you have licensed the Dynamic Data Access module.

9. On the **Production Data Source** tab, complete one of the following sets of steps to specify the location of the data source to use during production:

| To | Do this |
|--|---|
| Use data from an external file | <ol style="list-style-type: none">From the Type list, select File.In the File to use in production box, enter the symbolic name of the external file used to provide data. When you run the engine, you use the FILEMAP engine switch in a control file to map this symbolic file name to a physical path and file name that is valid for the production platform. <p>Note: On a z/OS platform, it is good practice to precede the symbolic name with DD:, and the remaining portion of the name must be eight or fewer characters.</p> <p>For more information about the FILEMAP engine switch, see <i>Switch Reference</i> in the Exstream Design and Production documentation.</p> |
| Use data from an external user-defined routine that is identified by a connector object in the Library | <ol style="list-style-type: none">From the Type list, select Connector.In the Connector box, select an existing connector object from the Library. <p>Note: The Connector option is available only if you have licensed the Dynamic Data Access module.</p> |

10. From the **Edit** menu, select **Save**.

6.2.5 Mapping a JSON Formatted Data File

You can map a JSON formatted data file in the Edit Panel using the JSON automapping feature, or you can map the data file manually. Automapping lets you automatically find areas of a data file, identify the type of data in the area, and map a variable. Manual mapping includes tasks that you must complete for all data files, such as defining the start of a customer, whether the data files are automapped or manually mapped. If you must map variables with names that do not match the element names, you must use manual mapping.

After you map a data area, Design Manager highlights the data area in a different color. If you want information about what the highlighting colors mean in the Edit Panel, right-click the Edit Panel and select **View > What do colors mean**.

Keep in mind that you are not required to map variables to all of the elements in a data file. The engine stores all of the variable values in memory during processing. To make processing faster and to reduce the amount of memory required, map only those elements you want to use to populate variables.

To map a JSON formatted data file, complete the following tasks as needed:

- “[Automapping a JSON Formatted Data File](#)” on the next page
- “[Manually Mapping a JSON Formatted Data File](#)” on page 263

- “[Viewing the Data Mapping Layout](#)” on page 271

Automapping a JSON Formatted Data File

You can use automapping to automatically map element or array values. If you have JSON data prepared for mapping and have not yet created variables, you can also create variables based on information in elements. If you are creating variables during automapping, Design Manager reads the file and selects the basic name, data type, and format features of a variable based on the data source. Although minor modifications might be required to change the default selection for a particular element, or to define properties such as valid values or output format, automapping is usually faster than manual mapping. After variables are created, or if you are using existing variables, Design Manager maps them to the data automatically.

You can use the automapping feature to map the data file using new variables created as part of the automapping process, or using existing variables. If you are creating new variables, Design Manager uses the element or array names to name the variables and to define the properties on the **JSON Auto-Map** dialog box. If an array does not have a name, the mapped variable is named root. All other variable and data area properties are set to the default. If you want to use existing variables to automap a JSON formatted data file, the variable names and properties must match the variable names and properties specified in the **JSON Auto-Map** dialog box.

To automap a JSON formatted data file:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Click the **Basic** tab.
3. From the **Data mapping method** list, select **Sample file (manual)**.
4. Depending on when you want to complete the automapping, complete one of the following sets of steps to automap the data file:

| To | Do this |
|---|--|
| Automap the data file during data file creation | On the Basic tab, click  . The JSON Auto-Map dialog box opens. |
| Automap the data file after it has been created | a. From the Library, drag the data file to the Edit Panel. b. Right-click any empty area and select Element > Auto-Map . The JSON Auto-Map dialog box opens. The JSON Auto-Map dialog box opens. |

5. In the **Folder** box, select the folder in the Library in which to store new variables or to find existing variables.
6. If you have one or more data areas with existing maps and you want to remove the maps before automapping, select the **Remove all existing data maps** check box. Clear the check box if you do not have existing data maps or if you want to double-map the data

areas with existing maps.

7. To name the variable, do one or more of the following:

| To | Do this |
|---|--|
| Change the name of the variable | <p>In the Name box, enter the name you want to use for the variable.</p> <p>By default, the element name or the attribute value defines the variable name. If the element name value has a space, Design Manager substitutes an underscore for the space.</p> |
| Add a prefix to all variable names | <ol style="list-style-type: none">a. In the Variable name prefix box, enter a string of characters.b. Click Apply. |
| Add the parent element name to resolve duplicate variable names | <p>Click Resolve using parent names.</p> <p>Note: If you have duplicate element names, keep in mind that the order in which you add the prefix affects the final variable name. If you resolve the duplicate names using parent names before you add a prefix, Design Manager adds the prefix in front of the parent name. However, if you add the prefix before resolving the duplicates, Design Manager adds the parent name in front of the prefix.</p> |

8. To update the properties of a variable, complete the following steps:
 - a. Click the element in the hierarchy tree area.
 - b. From the **Type** list, select a data type.
 - c. If the variable is an array, select the **Array** check box. Design Manager does not detect whether a variable should be an array. If the variable can have more than one value for a customer, you must select the **Array** check box.
 - d. From the **Format** list, select an output format. The options vary depending on your selection from the **Type** list.

For more information about variable types and output formats, see “[Types of Variables](#)” on [page 60](#) and “[Formatting Variable Values in Output](#)” on [page 81](#).

9. To exclude specific elements from automapping, complete one of the following sets of steps:

| To | Do this |
|--|---|
| Exclude a specific element from automapping | <ol style="list-style-type: none">a. Click the element in the hierarchy tree area.b. Select the Ignore check box. |
| Exclude all elements except the selected element and all its nested elements | <ol style="list-style-type: none">a. Click the parent element in the hierarchy tree area.b. Click Ignore all but this branch. |

| To | Do this |
|---|---------------------------------------|
| Clear ignore flags and include all of the ignored elements in automapping | Click Clear all ignore flags . |

10. If you are automapping a report file, complete the following steps to define the action to take when data is missing:
 - a. To define the action you want the engine to take when it encounters a missing value, select one of the following options from the **Action on missing value** list:
 - **Output empty element**—Include the element in the report and assign it a null value.
 - **SUPPRESS ELEMENT**—Do not include the element in the report.
 - **THROW AN ERROR**—Do not include the element in the report and issue an error message.
 - b. To apply the option you selected from the **Action on missing value** list to all of the elements in the data file, click **Apply to all**.
 - c. To apply the option you selected from the **Action on missing value** list if the engine encounters a zero or a blank in the element value, select the **Interpret zero or blank element value as missing data** check box.
 - d. To interpret zero or blank element values in all of the elements in the data file as missing data, click **All**.
11. Click **OK**.

Design Manager automaps the data file and highlights the data areas based on their use to indicate that you have mapped values to the data areas. In addition, you can view information (such as the name) about the variable when you place your pointer over the data area.

Manually Mapping a JSON Formatted Data File

You can manually map all of the JSON formatted data file types. Manual mapping is especially useful when you must use variable names that are different from the JSON element names or when you want to customize the options that are set during automapping.

For more information about automatically mapping a JSON formatted data file, see [“Automapping a JSON Formatted Data File” on page 261](#).

This section discusses the following topics:

- [“Identifying Customers in JSON Data” on the next page](#)
- [“Identifying Data Sections in JSON Data” on the next page](#)

- “[Identifying Recipients in JSON Data](#)” on the next page
- “[Manually Mapping a Variable to an Element Name or Value in JSON Data](#)” on page 266
- “[Specifying When the Engine Writes Data to an Element in a Report File](#)” on page 270

Identifying Customers in JSON Data

For JSON data sources, you must specify which JSON element starts a new customer. You must identify at least one JSON element as a customer in the data; otherwise, the engine will not process your application.

To identify a customer in JSON data:

1. In Design Manager, from the Library, drag the JSON data file to the Edit Panel.
 2. In the data, select the start of the unmapped element or array that represents the customer.
 3. Right-click the highlighted data and select **Element > Properties**.
- The **JSON Element Mapping Properties** dialog box opens.
4. Click the **Options** tab.
 5. To select where the new customer mapping begins in the data, do one of the following:

| To | Do this |
|--|---|
| Start a new customer before the selected element | From the Starts customer list, select At start of element . |
| Start a new customer after the selected element | From the Start customer list, select At end of element . |

Note: By default, the **Starts customer** list is set to **None**. This setting indicates that the selected element does not start a customer.

6. Click **OK**.

The **JSON Element Mapping Properties** dialog box closes and a solid black line appears above each instance of the element to indicate the start of a new customer. Each customer ends when the engine encounters the next customer element in the JSON data.

7. From the Menu bar, select **Edit > Save**.

Identifying Data Sections in JSON Data

For JSON data sources, you can specify which JSON element starts a new data section. If you do not identify the start data section element, the engine does not reset the variables, which might cause incorrect results at run time.

For more information about section data, see “[Section Data in Exstream Design and Production](#)” on page 4.

To identify a data section in JSON data:

1. In Design Manager, from the Library, drag the data file to the Edit Panel.
 2. In the data, select the start of the unmapped data section element or data section array.
 3. Right-click the highlighted data and select **Element > Properties**.
- The **JSON Element Mapping Properties** dialog box opens.
4. Click the **Options** tab.
 5. To specify where the new data section mapping begins in the data, do one of the following:

| To | Do this |
|---|---|
| Always start a new data section at the selected element | <ol style="list-style-type: none">a. From the Starts section list, select Always.b. In the Section box, enter the name of the data section. |
| Start a new data section only at the first occurrence of the selected element | <ol style="list-style-type: none">a. From the Starts section list, select First.b. In the Section box, enter the name of the data section. |

Note: By default, the **Starts section** list is set to **None**. This setting indicates that the selected element does not start a data section.

6. Click **OK**.

The **JSON Element Mapping Properties** dialog box closes and a solid red line appears above each instance of the data section element to indicate the start of a new data section.

Note: By default, the **End section(s) or recipient(s) at closing element** check box on the **Advanced** tab of the JSON data file properties is selected. This setting indicates that data sections and recipients end at the end element for the data section or recipient.

7. From the Menu bar, select **Edit > Save**.

Identifying Recipients in JSON Data

When you are using recipient data (data for recipients of additional copies of documents), you must identify where a new recipient starts. If you do not identify the start recipient element when using recipient data, the engine does not reset the variables, which might cause incorrect results at run time.

For more information about adding recipient copies of documents, see *Designing Customer Communications* in the Exstream Design and Production documentation.

To identify a recipient in the JSON data:

1. In Design Manager, from the Library, drag the data file to the Edit Panel.
2. In the data, select the start of any unmapped recipient element or recipient array.
3. Right-click the highlighted data and select **Element > Properties**.
The **JSON Element Mapping Properties** dialog box opens.
4. Click the **Options** tab.
5. To start a new recipient at the selected element, select **Always** from the **Starts recipient** list.

Note: By default, the **Starts recipient** list is set to **None**. This setting indicates that the selected element does not start a recipient.

6. Click **OK**.

The **JSON Element Mapping Properties** dialog box closes. A solid green line appears above each instance of the recipient element to indicate the start of a new recipient.

Note: By default, the **End section(s) or recipient(s) at closing element** check box on the **Advanced** tab of the JSON data file properties is selected. This setting indicates that data sections and recipients end at the end element for the data section or recipient.

7. From the Menu bar, select **Edit > Save**.

Manually Mapping a Variable to an Element Name or Value in JSON Data

1. In Design Manager, from the Library, drag the JSON data file to the Edit Panel.
2. In the data, select any unmapped element or array.
3. Right-click the highlighted data and select **Element > Properties**.
The **JSON Element Mapping Properties** dialog box opens.
4. Click the **Basic** tab.
5. To map the element, do one of the following:

| To | Do this |
|--|--|
| Map an existing variable to the element name | <ol style="list-style-type: none">Click the Data mapped to element name box. A prompt opens, asking whether you want to use an existing variable.Click Yes. The Select Variable dialog box opens.Select the variable from the list.Click OK. The Data Area Properties dialog box opens. |
| Map a new variable to the element name | <ol style="list-style-type: none">Click the Data mapped to element name box. A prompt opens, asking whether you want to use an existing variable.Click No. The New Variable dialog box opens.In the Name box, enter a name. In the Description box, enter a description (optional).From the Type list, select the type of data that the variable represents.If the variable can contain more than one value, select the Array check box.In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples.Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Map an existing variable to an element value | <ol style="list-style-type: none">Click the Data mapped to element value box. A prompt opens, asking whether you want to use an existing variable.Click Yes. The Select Variable dialog box opens.Select the variable from the list.Click OK. The Data Area Properties dialog box opens. |

| To | Do this |
|---|---|
| Map a new variable to an element value | <ol style="list-style-type: none">Click the Data mapped to element value box. A prompt opens, asking whether you want to use an existing variable.Click No. The New Variable dialog box opens.In the Name box, enter a name. In the Description box, enter a description (optional).From the Type list, select the type of data that the variable represents.If the variable can contain more than one value, select the Array check box.In the Design sample box, enter the text string that you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional). Exstream Design and Production does not support Unicode for design samples.Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. |
| Skip the selected element during processing | Select the Ignore this element check box. If you select this option, all of the other options on the dialog box are disabled. Skip to step 8. |

6. In the **Data Area Properties** dialog box, verify or update the properties for the data area.

For information about specifying how the engine uses a data area during processing, see “[Specifying How the Engine Uses the Data Area During Processing](#)” on page 314.

For information about specifying the input format of data, see “[Specifying the Input Format of Data in a Data Area](#)” on page 154.

7. Click **OK**.

The **Data Area Properties** dialog box closes.

8. On the **JSON Element Mapping Properties** dialog box, click the **Options** tab and select one or both of the following options to specify how the engine processes the data:

- To remove white space at the beginning or end of the element value, select the **Suppress whitespace** check box.
- To specify that the engine should apply a default value defined in the variable if an element is missing, select the **Use previous value if missing** check box.

9. If you are creating an auxiliary layout file or a report file, you must complete the following steps to specify how the engine writes out data from the data file:

- To define the action that you want the engine to take when it encounters a missing value:

- i. From the **Action on missing value** list, select one of the following options:
 - **Output empty element**—Include the element in the output file and assign it a null value.
 - **Suppress element**—Do not include the element in the output file.
 - **Throw an error**—Do not include the element in the output file and issue an error message.
- ii. If the engine should identify a zero or a blank element value as a missing value, select the **Interpret zero or blank element value as missing data** check box.

The action that you selected from the **Action on missing value** list is applied to any zero or blank value in the data.

Note: If you are using the data file with DLF output, make sure that you select the **Interpret zero or blank element value as missing data** check box. DLF files do not remove elements when information is removed; instead the DLF file leaves the element value blank.

- b. If you are mapping an array variable, in the **Multiplicity for arrays** area, complete the following steps as needed to specify how array elements increment missing elements with default values to keep arrays in sync within the JSON hierarchy:

| To | Do this |
|---|--|
| Prevent arrays from incrementing missing elements | In the Min value box, enter 0 . |
| Specify the minimum number of times to increment the element under the parent element | In the Min value box, enter the minimum number of times to repeat the element. |
| Specify a maximum number of times to increment the element under the parent element | <ol style="list-style-type: none">i. Clear the Unbounded max value check box.ii. In the Max value box, enter the maximum value. |
| Specify that there is no maximum number of times to increment the element | Select the Unbounded max value check box. |

- c. If you are mapping a report file and want to filter the output using a rule, enter a rule in the **Filter** box.

10. Click **OK**.

The **JSON Element Mapping Properties** dialog box closes and Design Manager applies a color to the element value to indicate that you have mapped the name to a variable. The color of the data mapping can vary depending on the settings that you apply. To view a reference for what the colors in the Edit panel represent, right-click anywhere in the Edit panel and select **View > What do colors mean**. In addition, you can also view information

about the mapping (such as the variable name) when you hover the pointer over a mapped data area.

For more information about data area properties, see [“Using a Data Area to Specify How the Engine Reads and Writes Data” on page 151](#).

Specifying When the Engine Writes Data to an Element in a Report File

You can change the engine timing so that the engine writes data to a report file at different times. You can define the engine timing either in the data file properties or in the element properties. When you want to apply the same timing to all of the elements in the data file, define the engine timing in the data file. When you want to apply specific timing to individual elements, define the engine timing in the properties for each element.

For information about defining the engine timing in a report file, see [“Defining the Records in a Report or Post-sort Report File” on page 55](#).

To specify when the engine writes data to an element in a report file:

1. In Design Manager, from the Library, drag a report file to the Property Panel.
2. Click the **Advanced** tab.
3. From the **IO time** list, select **As determined by record properties**.
4. From the **File** menu, select **Save**.
5. From the Library, drag the report file to the Edit Panel.
6. Highlight any unmapped element name or value, right-click, and select **Element > Properties**.

The **JSON Element Mapping Properties** dialog box opens.

7. Click the **Basic** tab.
8. Map an element name or value.
9. Click the **Options** tab.
10. From the **IO time** list, select an option to specify when the engine reads data records.
11. Click **OK**.

The **JSON Element Mapping Properties** dialog box closes and Design Manager highlights the element value, based on its use, to indicate that you have mapped the name to a variable. In addition, you can view information about the variable (such as the name) when you place your pointer over the data area.

Viewing the Data Mapping Layout

You can view details about the data mapping layout of a data file. For example, you can review the properties of all of the variables that are mapped to the data file.

To view the data mapping layout:

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click and select **View > View Layouts**.

A list of the tags or elements in the layout opens in the Property Panel.

3. Click a specific tag or element, and if it is present in the mapped data file, it is highlighted in the Edit Panel.

6.2.6 Defining the Structure of a JSON Formatted Output Data File

If you are creating an auxiliary layout file or a report file, you can use the data file properties to define the structure of the data the engine creates.

To define the structure of a JSON formatted output data file:

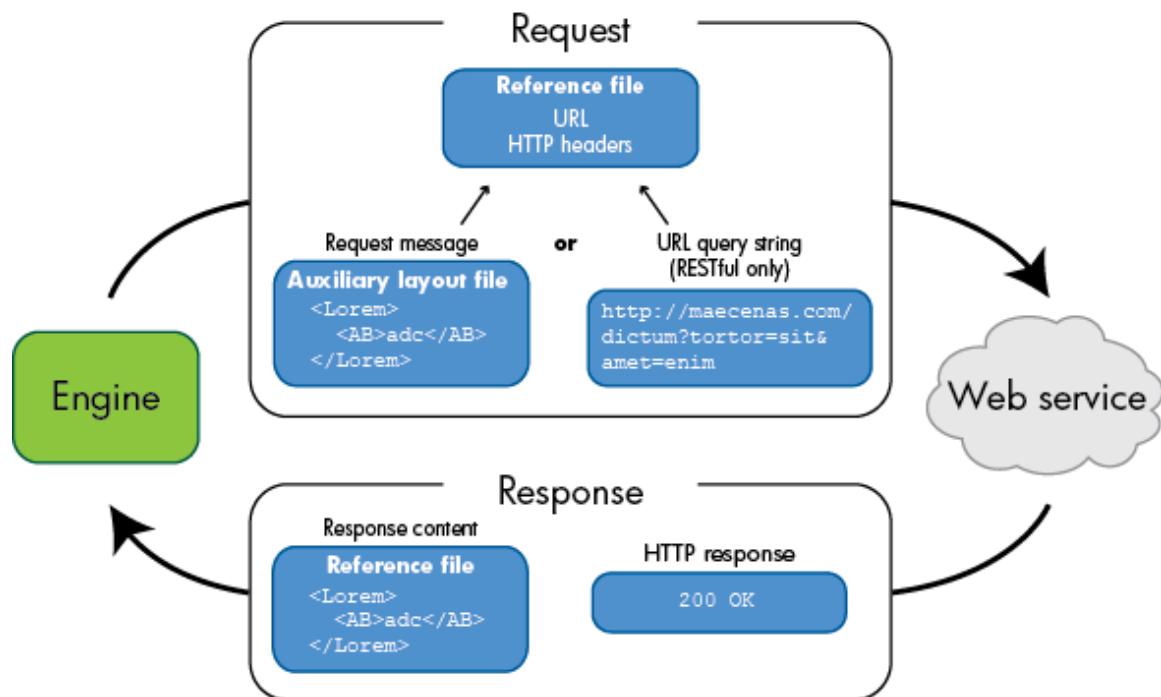
1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. From the **Encoding** list, select the encoding you want to include in the declaration tag. Select the option that most closely matches your native code page, or enter your own encoding. If you do not want to declare an encoding, leave this option blank.
3. In the **Indentation for each tag level** box, enter the number of spaces (from 0 to 10) to indent for each tag level in the JSON output.
4. To suppress indentation and carriage returns or line feeds in JSON output in order to reduce the size of the file, select the **Lean JSON** check box.
5. From the **Edit** menu, select **Save**.

Chapter 7: Setting Up Data Files to Communicate with a Web Service

Web service data files let you create output using data delivered by a web service during production. You can also use a web service data file to store requested data from a web service. You can use web service data files without additional web service setup on the target production platform. If you are working in a Service-Oriented Architecture (SOA) based organization, you can also leverage your existing application infrastructure. Web service data files are available only if you have licensed the Web Services Interface, Empower, or DLF modules. Exstream supports SOAP and Representational State Transfer (REST) for communication with web services using data files.

As shown in the following example, Exstream sends data to and receives data from a web service using XML or JSON formatted web service data files.

Web service data file interaction



To request data from a web service, you can use an auxiliary layout file to contain the request message. When using a RESTful web service, the auxiliary layout file is optional; you can also directly request data using only the URL. You use a reference file to receive the data from the web service, and you specify the server URL and define the format of the response from the server in the properties of the reference file. When you use an auxiliary layout file for the request message, the engine uses the HTTP POST (SOAP or REST) or PUT (REST only) method to send the request data. If you use only a URL for a RESTful web service and no auxiliary layout

file, the engine uses the GET method to directly request data. If a request is successful, the web service sends the requested data back to the engine, and the engine parses the data and stores it in the reference file. If the request is unsuccessful, the web service sends an HTTP error response and, when using a SOAP web service, a SOAP fault to the engine.

When you use only the URL to send a request to a RESTful web service, you can also use an initialization file instead of a reference file to receive data from the web service. You cannot specify an auxiliary layout file in an initialization file, so you cannot use an initialization file to receive data when using an auxiliary layout file to contain the request message.

To upload data to a web service, you use a report file or post-sort report file to contain the data to be uploaded. You can choose whether to use the HTTP POST or PUT method.

Keep in mind that you can use XML or JSON formatted data files to communicate with a RESTful web service. However, you must use XML formatted data files if you want to communicate with a SOAP web service.

For more information about setting up your data files, see the following:

- [“Setting Up XML Formatted Data Files” on page 178](#)
- [“Setting Up JSON Formatted Data Files” on page 253](#)

If you are using data files to communicate with a web service in a Windows production environment and you do not plan to install the design environment, you must install the Microsoft Visual C++ 2013 Redistributable Package.

For more information about installing the production environment on Windows, see *Installation and Upgrade Information* in the Exstream Design and Production documentation.

Caution: If you implemented the Web Services Interface module in a version of Exstream Design and Production earlier than 7.0, you must replace the content output files with auxiliary layout files and reference files. Content output data files are not supported in Exstream Design and Production 7.0 and later.

To set up data files to communicate with a web service, complete the following tasks as needed:

- [“Using an Auxiliary Layout File to Define a Web Service Request Message” on the next page](#)
- [“Using a Reference File or an Initialization File to Define a Request and Receive Data” on page 275](#)
- [“Using a Report File or a Post-Sort Report File to Upload Data” on page 282](#)

7.1 Using an Auxiliary Layout File to Define a Web Service Request Message

Auxiliary layout files use JSON or XML to format the request message that the engine sends to a web service during a request. The engine creates a web service request using the structure in the auxiliary layout file. In addition, for a SOAP web service, the customer start tag that you define in the auxiliary layout file becomes the customer key ID that the web service uses to look up data. Since the timing of the data lookup is often important, keep in mind that the engine keys data files before it populates the data in the design and before it creates pages.

Note: You must use XML formatted data files if you want to communicate with a SOAP web service.

The engine sends data to a web service based on the timing of events during engine processing. For example, suppose that as the engine reads a data file, it reads a data section that is set up to trigger the engine to search for additional data on a web service. When the engine reads the data section, it uses the auxiliary layout file to format the request that it then sends to the web service. Use the Trigger function to define a custom timing for when to use the auxiliary layout file and, consequently, when to send a web service request. This function lets you create a custom timing for operations, such as sending request data to a web service, during engine processing.

For more information about the Trigger function, see *Using Logic to Drive an Application* in the Exstream Design and Production documentation.

To request data from a web service using an auxiliary layout file, you must also use a reference file. The reference file defines the URL of the web service and the format of the response from the web service. You must include the reference file in an application, but the inclusion of the auxiliary layout file is optional.

For more information about setting up a reference file for use with a web service, see “[Using a Reference File or an Initialization File to Define a Request and Receive Data](#)” on the next page.

For more information about additional data file properties, see “[Creating an Auxiliary Layout Data File](#)” on page 18 and “[Setting Up XML Formatted Data Files](#)” on page 178.

To use an auxiliary layout file to define a web service request message:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.
The **New Data File** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Auxiliary layout**.

4. From the **File format** list, select **JSON** or **XML**.
5. From the **File source type** list, select **Web Service (SOAP)** or **Web Service (RESTful)**, depending on whether you are using a SOAP or RESTful Web Service.

Note: The Web Service (SOAP) source is not available for the JSON file format.

6. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

7. Set the data file properties as necessary.
8. From the **File** menu, select **Save**.
9. From the Library, drag the auxiliary layout file to the Edit Panel.
10. Map the file as necessary.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.

11. From the **File** menu, select **Save**.

7.2 Using a Reference File or an Initialization File to Define a Request and Receive Data

Any time you request data from either a SOAP or RESTful web service, you must use a reference file to define the request and receive the data, or, if you are using only a URL to request data from a RESTful web service, you can use an initialization file instead of a reference file. The reference file or initialization file defines the URL of the web service and the format of the response from the web service. When you use an auxiliary reference file to define a request message, the reference file also specifies the auxiliary file to use. You must include the reference file in an application, but the inclusion of the auxiliary layout file is optional. You cannot use an auxiliary layout file with an initialization file.

For related information about setting up and using data files to define web service requests, see the following topics:

- “[Creating a Reference Data File](#)” on page 37
- “[Setting Up JSON Formatted Data Files](#)” on page 253
- “[Setting Up XML Formatted Data Files](#)” on page 178
- “[Using an Auxiliary Layout File to Define a Web Service Request Message](#)” on the previous page

- “[Using a Reference File or an Initialization File to Define a Request and Receive Data from a RESTful Web Service](#)” on page 279
- “[Using a Reference File to Define a Request and Receive Data from a SOAP Web Service](#)” below

7.2.1 Using a Reference File to Define a Request and Receive Data from a SOAP Web Service

When communicating with a SOAP web service, you must use an auxiliary layout file to define the request message. The reference file then specifies the auxiliary file to use. You must include the reference file in an application, but the inclusion of the auxiliary layout file is optional.

When the SOAP web service performs a successful lookup, it responds with an HTTP status of 200 OK and returns the requested data. The engine then uses the reference file to parse and store the returned data. If the web service cannot find the requested data or if there is another error, the web service sends a SOAP fault with an HTTP status of 500 Internal Server Error. The engine receives the SOAP fault and parses it automatically to provide you with troubleshooting information to help you correct the error. The following table summarizes how the engine parses the response based on the HTTP response code.

Engine parsing based on HTTP response when using SOAP

| HTTP Response | Engine parsing method |
|---------------------------|--|
| 200 OK | The engine parses the returned data using the data map specified in the associated web service reference file. |
| 500 Internal Server Error | The engine automatically parses valid SOAP 1.1 or 1.2 faults sent to the engine. The engine automatically populates SOAP-related system variables with the appropriate SOAP fault information. For example, the engine places the SOAP fault code in the 'SYS_SoapFaultCode' variable. If you have mapped the associated web service reference file, the engine ignores the data map. For more information about the SOAP-related system variables, see “Real-time and SOAP-Related System Variables” on page 127 . |

To use a reference file to define a request and receive data from a SOAP web service:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.
The **New Data File** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Reference file**
4. From the **File format** list, select **XML**.
5. From the **File source type** list, select **Web Service (SOAP)**.
6. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

7. From the **Reference key layout** list, select the auxiliary layout file that sends the data to the web service. Exstream Design and Production supports auxiliary layout files up to 4MB. Keep in mind that the size of the auxiliary layout file can be larger than 4MB after DLF or content XML is inserted in the request.
8. In the **Data mapping source** box, enter the file path to the external file used to provide data during mapping.
9. If you want to include HTTP headers in the request, enter an array variable that contains the headers to use in the **HTTP headers** box.
10. If you want to receive the HTTP response status after the request, enter a variable to store the response status in the **HTTP response code** box.
11. If you want to use HTTP authentication, complete the following steps:
 - a. Select the **Use authentication** check box.
 - b. In the **User name** section, select a variable that contains the user name to use for authentication, or enter a static user name in the box.
 - c. In the **Password** section, select a variable that contains the password to use for authentication, or enter a static password in the box.
12. To define the web service that is used for testing and mapping, complete the following steps:
 - a. Click the **Test Data Source** tab.
 - b. From the **Type** list, select **SOAP**.
 - c. In the **URL** box, enter the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. In the **SOAPAction HTTP header** box, enter the Uniform Resource Indicator (URI) that identifies the intent of the SOAP HTTP request header field. Enter an empty string to indicate that the HTTP Request-URI provides the intent of the SOAP message.
 - e. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - f. In the **Transaction Options** area, select the radio button box that corresponds to the type of file transfer you want to use when sending files to a server:

- **SOAP 1.1 content specification (default)**
 - **SOAP 1.2 content specification**
13. To define the web service that is used for production, complete the following steps:
- a. Click the **Production Data Source** tab.
 - b. From the **Type** list, select **SOAP**.
 - c. In the **URL** box, enter the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. In the **SOAPAction HTTP header** box, enter the Uniform Resource Indicator (URI) that identifies the intent of the SOAP HTTP request header field. Enter an empty string to indicate that the HTTP Request-URI provides the intent of the SOAP message.
 - e. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - f. In the **Transaction Options** area, select the radio button box that corresponds to the type of file transfer you want to use when sending files to a server:
 - **SOAP 1.1 content specification (default)**
 - **SOAP 1.2 content specification**
14. From the **File** menu, select **Save**.
15. From the Library, drag the reference file to the Edit Panel.
16. Map the file as necessary.
- For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.
17. From the **File** menu, select **Save**.

Note: To change the information for a web service without repackaging your application, use the WEBSERVICEMAP switch in your control file. This switch lets you control the URL, SOAPAction HTTP header, and file transfer method.

For more information about the WEBSERVICEMAP switch, see *Switch Reference* in the Exstream Design and Production documentation.

7.2.2 Using a Reference File or an Initialization File to Define a Request and Receive Data from a RESTful Web Service

When using a RESTful web service, you can use an auxiliary layout file to define a request message, or you can use only the URL of the web service.

When using an auxiliary layout to define the request message for a RESTful web service, you use a reference file to specify the auxiliary file to use and whether to use the PUT or POST method to send the request message. You must include the reference file in an application, but the inclusion of the auxiliary layout file is optional.

If you do not use an auxiliary layout file, the engine uses the GET method to request data using only the URL you specify. Depending on the web service you are using, you can optionally include a query string in the web service URL specified in the reference file. Furthermore, if you do not need to specify an auxiliary layout file, you can use an initialization file instead of a reference file to define the request and receive data.

For more information about data file types, see [“Data File Types” on page 12](#).

RESTful web services do not natively provide the kind of detailed error information that is contained in a SOAP fault. When using REST, you might receive other HTTP error status codes, and any data returned might still be mapped to a reference file. You can use a combination of the specific error status and the returned data to troubleshoot the error.

To use a reference file or an initialization file to define a request and receive data from a RESTful web service:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. Depending on whether you are using an auxiliary layout file, perform one of the following actions:

| If you are using this method | Do this |
|------------------------------|---|
| Auxiliary layout file | Select Reference file from the File type list and then select XML or JSON from the File format list. |
| URL only | Do one of the following: <ul style="list-style-type: none">• Select Initialization file from the File type and then select XML or JSON from the File format list.• Select Reference file from the File type and then select XML or JSON from the File format list. |

4. From the **File source type** list, select **Web Service (RESTful)**.

5. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

6. Depending on whether you are using an auxiliary layout file, do one of the following:

- If you are using an auxiliary layout file, select from the **Reference key layout** list the auxiliary layout file that sends the data to the web service.
- If you are using only a URL, select **No Layout** from the **Reference key layout** list.

Exstream Design and Production supports auxiliary layout files up to 4MB. Keep in mind that the size of the auxiliary layout file can be larger than 4MB after the DLF or content XML or JSON is inserted in the request.

Note: The **Reference key layout** list does not appear for initialization files.

7. If you want to include HTTP headers in the request, enter an array variable that contains the headers to use in the **HTTP headers** box.
8. If you want to receive the HTTP response status after the request, enter a variable to store the response status in the **HTTP response code** box.
9. If you want to use HTTP authentication, complete the following steps:
 - a. Select the **Use authentication** check box.
 - b. In the **User name** section, select a variable that contains the user name to use for authentication, or enter a static user name in the box.
 - c. In the **Password** section, select a variable that contains the password to use for authentication, or enter a static password in the box.
10. To define the web service that is used for testing and mapping, complete the following steps:
 - a. Click the **Test Data Source** tab.
 - b. From the **Type** list, select **RESTful**.
 - c. In the **URL** box, enter the web address of the server, including any query string you want to use, or, in the variable box, enter a variable that contains the web address of the server, including any query string. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select

this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.

- e. If you are using an auxiliary layout file, select the radio button in the **Transaction Options** area that corresponds to the HTTP method you want to use for sending the request message:
 - **HTTP PUT**—Use the PUT method to send the request message.
 - **HTTP POST**—Use the POST method to send the request message.

If you are using only a URL, the HTTP method is restricted to **HTTP GET**.

11. To define the web service that is used for production, complete the following steps:
 - a. Click the **Production Data Source** tab.
 - b. From the **Type** list, select **RESTful**.
 - c. In the **URL** box, enter the web address of the server, or, in the variable box, enter a variable that contains the web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - e. If you are using an auxiliary layout file, select the radio button in the **Transaction Options** area that corresponds to the HTTP method you want to use for sending the request message:
 - **HTTP PUT**—Use the PUT method to send the request message.
 - **HTTP POST**—Use the POST method to send the request message.

If you are using only a URL, the HTTP method is restricted to **HTTP GET**.

12. From the **File** menu, select **Save**.
13. From the Library, drag the reference file or initialization file to the Edit Panel.
14. Map the file as necessary.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.
15. From the **File** menu, select **Save**.

Note: To change the URL for the web service called without repackaging your application, use the WEBSERVICEMAP switch in your control file.

For more information about the WEBSERVICEMAP switch, see *Switch Reference* in the Exstream Design and Production documentation.

7.3 Using a Report File or a Post-Sort Report File to Upload Data

Web service report files and post-sort report files let you upload XML or JSON formatted data to a web service. You can thus reuse the data created by Exstream Design and Production in subsequent processing. For example, you can update external systems, write reports, or initiate variable data processing.

Unlike reference files and auxiliary layout files, which must always be used together in order to communicate with a web service, report files and post-sort report files are optional. When you use a report file, the engine sends a request to the web service with the data to upload, and you can determine success or failure from the HTTP response.

For more information about additional data file properties, see the “[Creating a Report or a Post-sort Report File](#)” on page 50 and “[Setting Up XML Formatted Data Files](#)” on page 178.

7.3.1 Using a Report or Post-Sort Report File to Upload Data to a SOAP Web Service

To use a report file or a post-sort report file to upload data to a SOAP web service:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Report file** or **Post-sort report file**.
4. From the **File format** list, select **XML**.
5. From the **File source type** list, select **Web Service (SOAP)**.
6. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

7. In the **Data mapping source** box, enter the file path to the external file used to provide

data during mapping.

8. If you want to include HTTP headers in the request, enter an array variable that contains the headers to use in the **HTTP headers** box.
9. If you want to receive the HTTP response status after the upload request, enter a variable to store the response status in the **HTTP response code** box.
10. If you want to use HTTP authentication, complete the following steps:
 - a. Select the **Use authentication** check box.
 - b. In the **User name** section, select a variable that contains the user name to use for authentication, or enter a static user name in the box.
 - c. In the **Password** section, select a variable that contains the password to use for authentication, or enter a static password in the box.
11. To define the web service that is used for testing and mapping, complete the following steps:
 - a. Click the **Test Data Source** tab.
 - b. From the **Type** list, select **Web service**.
 - c. In the **URL** box, enter the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. In the **SOAPAction HTTP header** box, enter the Uniform Resource Indicator (URI) that identifies the intent of the SOAP HTTP request header field. Enter an empty string to indicate that the HTTP Request-URI provides the intent of the SOAP message.
 - e. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - f. In the **Transaction Options** area, select the check box that corresponds to the type of file transfer you want to use when sending files to a server:
 - **SOAP 1.1 content specification (default)**
 - **SOAP 1.2 content specification**
12. To define the web service that is used for production, complete the following steps:
 - a. Click the **Production Data Source** tab.
 - b. From the **Type** list, select **Web service**.
 - c. In the **URL** box, enter the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is

- HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
- d. In the **SOAPAction HTTP header** box, enter the Uniform Resource Indicator (URI) that identifies the intent of the SOAP HTTP request header field. Enter an empty string to indicate that the HTTP Request-URI provides the intent of the SOAP message.
 - e. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - f. In the **Transaction Options** area, select the check box that corresponds to the type of file transfer you want to use when sending files to a server:
 - **SOAP 1.1 content specification (default)**
 - **SOAP 1.2 content specification**
13. From the **File** menu, select **Save**.
14. From the Library, drag the reference file to the Edit Panel.
15. Map the file as necessary.
- For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.
16. From the **File** menu, select **Save**.

Note: To change the information for the web service called without repackaging your application, use the WEBSERVICEMAP switch in your control file. This switch lets you control the URL, SOAPAction HTTP header, and file transfer method.

For more information about the WEBSERVICEMAP switch, see *Switch Reference* in the Exstream Design and Production documentation.

7.3.2 Using a Report or Post-Sort Report File to Upload Data to a RESTful Web Service

To use a report file or a post-sort report file to upload data to a RESTful web service:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

The **New Data File** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Report file** or **Post-sort report file**.

4. From the **File format** list, select **XML or JSON**.

Note: The JSON file format is not available for post-sort report files.

5. From the **File source type** list, select **Web Service (RESTful)**.

6. Click **Finish**.

The **New Data File** dialog box closes and the data file opens in the Property Panel for you to define.

7. If you want to include HTTP headers in the request, enter an array variable that contains the headers to use in the **HTTP headers** box.

8. If you want to receive the HTTP response status after the upload request, enter a variable to store the response status in the **HTTP response code** box.

9. If you want to use HTTP authentication, complete the following steps:

- a. Select the **Use authentication** check box.
- b. In the **User name** section, select a variable that contains the user name to use for authentication, or enter a static user name in the box.
- c. In the **Password** section, select a variable that contains the password to use for authentication, or enter a static password in the box.

10. To define the web service that is used for testing and mapping, complete the following steps:

- a. Click the **Test Data Source** tab.
- b. From the **Type** list, select **RESTful**.
- c. In the **URL** box, enter the Web address of the server, or, in the variable box, enter a variable that contains the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
- d. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box in the **Transaction Options** area. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
- e. In the **Transaction Options** area, select the radio button that corresponds to the HTTP method you want to use for sending the request message:
 - **HTTP PUT**—Use the PUT method to send the data.
 - **HTTP POST**—Use the POST method to send the data.

11. To define the web service that is used for production, complete the following steps:

- a. Click the **Production Data Source** tab.
 - b. From the **Type** list, select **RESTful**.
 - c. In the **URL** box, enter the Web address of the server, or, in the variable box, enter a variable that contains the Web address of the server. By default, Exstream Design and Production uses HTTP protocol and port 80. If you specify that the protocol is HTTPS but do not specify a port, Exstream Design and Production defaults to port 443.
 - d. If you want to make sure the HTTPS protocol is used, select the **Enforce Secure Transaction (HTTPS)** check box. When you select this option, the engine uses the HTTPS protocol even if the URL specifies HTTP.
 - e. In the **Transaction Options** area, select the radio button that corresponds to the HTTP method you want to use for sending the request message:
 - **HTTP PUT**—Use the PUT method to send the data.
 - **HTTP POST**—Use the POST method to send the data.
12. From the **File** menu, select **Save**.
 13. From the Library, drag the reference file to the Edit Panel.
 14. Map the file as necessary.

For more information about mapping a data file, see “[Data File Mapping](#)” on page 129.
 15. From the **File** menu, select **Save**.

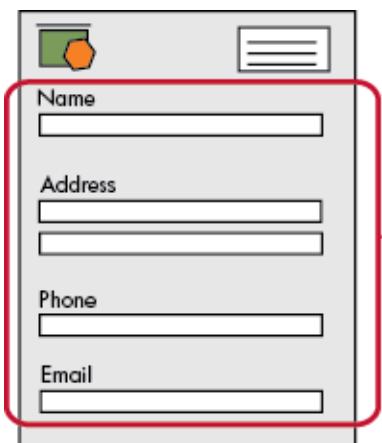
Note: To change the URL for the web service called without repackaging your application, use the WEBSERVICEMAP switch in your control file.

For more information about the WEBSERVICEMAP switch, see *Switch Reference* in the Exstream Design and Production documentation.

Chapter 8: Setting Up Data Files to Pre-Fill and Mine PDF Forms

Many enterprise processes rely on PDF forms to gather information from customers. If you use PDF XFA forms, you can integrate the forms into processes that use Exstream Design and Production. XML Forms Architecture (XFA) is an XML language that defines the design and data presentation of a PDF form. XFA lets you design a form and then capture customer data as XML tag values when customers complete the form. XFA handles the design and the data as separate entities. Because PDF form data files are in XML format, Exstream Design and Production can leverage the XML formatting and the data capabilities of XFA to let you interact with the data in a PDF XFA form.

XML data in a PDF XFA form



```
<?xml version="1.0" encoding="UTF-8"?>
<xfa:CustomerData xmlns:xfa="http://www.example.com/schema/xfa/">
  <CustomerForm 1>
    <Name>Sample</Name>
    <Address1>123 Sample Street</Address1>
    <Address2>City, ST 12345</Address2>
    <Phone>123-456-7890</Phone>
    <Email>Sample@Example.com</Email>
  </CustomerForm 1>
</xfa:CustomerData>
```

Exstream Design and Production lets you use existing data to pre-fill some or all of the fields in a PDF form. You can also use data contained in a PDF form to drive the customization and creation of applications just as you would with other data file formats. If desired, you can even "round-trip" the data. For example, you can use data extracted from one PDF form to pre-fill portions of a second form, or you can pre-fill a form and then leverage the completed form to drive the production of an application.

To help you set up data files to pre-fill or mine PDF forms, this chapter discusses the following topics:

- “[PDF Form Requirements and Considerations](#)” on the next page
- “[Pre-Filling Data Fields in PDF Forms](#)” on page 289
- “[Mining Data from PDF Form Fields](#)” on page 292

8.1 PDF Form Requirements and Considerations

As with other data file formats, before you set up a PDF form data file, make sure that you have licensed the appropriate Exstream Design and Production modules and that you consider the requirements and limitations of using PDF forms in Exstream Design and Production.

This section discusses the following topics:

- “[Module Requirements](#)” below
- “[PDF Form Considerations](#)” below

8.1.1 Module Requirements

Although PDF form data files are in XML format, you are not required to license the XML/JSON Input module. However, because XFA and Exstream Design and Production handle the design and data as separate entities, you must license the Dynamic Content Import module in order to pass through the XFA data. If you want to use PDF form data files to integrate PDF XFA forms with Exstream Design and Production applications, you must license one or both the following modules (in addition to the Dynamic Content Import module), depending on your business needs:

PDF XFA pre-filling and mining module requirements

| Module | Related information |
|-------------------|---|
| PDF Form Pre-fill | “Pre-Filling Data Fields in PDF Forms” on the next page |
| PDF Form Miner | “Mining Data from PDF Form Fields” on page 292 |

Note: The PDF Form Pre-Fill and PDF Form Miner modules are not supported on z/OS.

8.1.2 PDF Form Considerations

Before pre-filling or mining a PDF form, keep the following considerations in mind:

- PDF forms must be XFA version 2.2, which can be generated using Adobe Pro with LiveCycle version 7.0 or later. Forms that use the older PDF Acroform format are not supported.
- An application can include only one PDF form, but the form can have multiple pages. If the

PDF form has multiple pages, every page must be included, not just a specific page or range of pages.

- You can pre-fill multiple copies of the same form. When including multiple copies of the same PDF form in the same output file, subforms within the form must contain dynamic indexes in order to be populated correctly.
- You cannot resize, rotate, reposition, skew, or scale PDF forms.
- If you want pre-filled PDF forms to be editable when sent to customers, you must create PDF output.
- You cannot use Exstream Design and Production to create new PDF forms or to control the security features available in PDF forms. You must use the original PDF form design tool to create forms and to enable security features.
- Exstream Design and Production does not interact with the design of an existing PDF form. When Exstream Design and Production processes the form, the design is passed through to the engine unchanged. If you want to create an interactive form using Exstream Design and Production, the interactive (Live) capabilities provide extensive functionality.

For more information about the interactive (Live) capabilities of Exstream Design and Production, see *Designing for LiveEditor* in the Exstream Design and Production documentation.

8.2 Pre-Filling Data Fields in PDF Forms

If you use PDF XFA forms to collect information from customers, you can create a PDF form data file that lets you repurpose existing data to populate some of the fields. Pre-filling fields lets you make the form easier for customers to complete since they are not required to enter information that you already have in your systems, such as names and addresses. Since the fields are pre-filled and remain editable, customers can also quickly verify the accuracy of the data and update the data if necessary. For example, you can use data from a delimited data file to populate data in a PDF form so customers are required to complete only the empty form fields.

PDF form pre-fill example

The screenshot shows a PDF form with several fields. On the left, there is a table of data with rows labeled G, R, C, R, M. A red dashed box highlights the first row (G). An orange box highlights the entire table. On the right, there is a form with fields: Name (Mildred James), Address (two empty lines), Phone (one empty line), and Email (james M@email.com). The 'Name' field has a red arrow pointing to it, and the 'Email' field has an orange arrow pointing to it.

| | |
|---|---|
| G | 3/2/2010, Ab Ovo Usque Ad Mala, 117.47, Sava |
| G | 3/27/2010, Post Festum, 8.94, Savannah, GA,,, |
| C | Mildred, James, james M@email.com, 1812 East |
| R | 3/1/2010, Post Cibum, 18.56, Savannah, GA,,, |
| R | 3/12/2010, Quod Me Nutrit Me Destructit, 35.1 |
| M | 3/4/2010, Caveat Emptor, 109.62, Savannah, GA |

Keep in mind that XFA handles the design and data as separate entities, so Exstream Design and Production updates only the data elements in the XML. You cannot use Exstream Design and Production to edit the design of the form itself or to create a new form. During processing, Exstream Design and Production passes through the data from an input data file to the form using a placeholder variable. The engine then populates tag values in the XML according to variables mapped in a PDF form auxiliary layout file. The PDF form is then produced with the pre-filled fields. If you want the form to remain editable, you must create PDF output. The data is placed in the fields, but the form itself is not changed because the engine treats the form as an image.

For more information about creating PDF output, see *Creating Output* in the Exstream Design and Production documentation.

To pre-fill data fields in PDF forms, you must complete the following tasks:

1. [“Specifying the Data to Pre-Fill in a PDF Form” below](#)
2. [“Placing the Data on a PDF Form in Output” on the next page](#)

8.2.1 Specifying the Data to Pre-Fill in a PDF Form

To specify the data that you want to pre-fill in a PDF form, you must create a PDF form auxiliary layout file. The auxiliary layout file lets you use the layout of the data elements in the PDF XFA form to associate variables with the tag names and tag values in the form. At run time, the engine reads the data elements in the PDF XFA form and uses the variable mapping to pre-fill the form fields.

For more information about additional data file properties, see [“Creating an Auxiliary Layout Data File” on page 18](#).

To specify the data you want to pre-fill in a PDF form:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Auxiliary layout**
4. From the **File format** list, select **PDF Form**.
5. Click **Finish**.

The data file opens in the Property Panel.

6. Click the **Basic** tab.
7. In the **Sample layout** box, select the PDF XFA form used to map the variables.
8. From the Library, drag the auxiliary layout file to the Edit Panel.
9. Map variables to the form fields you want to pre-fill with customer data. You can use the same mapping features that are available for other XML formatted data files.
10. From the **File** menu, select **Save**.

For more information about mapping XML formatted data files, see “[Mapping an XML Formatted Data File](#)” on page 189.

8.2.2 Placing the Data on a PDF Form in Output

After you have defined the data you want to pre-fill, you must specify where the engine places that data in the PDF form. To place the data in a PDF form, use a placeholder variable. The placeholder variable takes the data from an input data file and passes it through to the PDF form using the layout you defined in the auxiliary layout file. Make sure that the placeholder variable references the auxiliary layout file that defines the data in the PDF XFA form.

For more information about additional placeholder variable properties, see *Importing External Content* in the Exstream Design and Production documentation.

To place the data on a PDF form in output:

1. In Design Manager, in the Library, right-click the **Data Dictionary** heading and select **New Variable**.

The **New Variable** dialog box opens.

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **Type** list, select **Placeholder**.
4. From the **Placeholder Type** list, select **PDF**.

5. If you are using multiple copies of the same form and the variable is an array, select the **Array** check box.
 6. Click **Finish**.
- The variable opens in the Property Panel.
7. Go to the **Placeholder** tab.
 8. If the file is a multi-page file, select **All** from the **Pages to import** list.
 9. From the **Auxiliary PDF form layout** box, select the auxiliary layout file that defines the data to pre-fill.
 10. From the **File** menu, select **Save**.

8.3 Mining Data from PDF Form Fields

If you use PDF XFA forms to collect information from customers, you can create a PDF form data file that lets you use the information you receive from customers as input in an Exstream Design and Production application. The engine uses the data, identified by the `<xfa:data>` tag, in the same way as it does other XML formatted data files. For example, suppose a customer returns a completed enrollment form. You can use the information the customer provided to drive the creation of a welcome packet that includes only information relevant to the customer.

PDF form mining example



For more information about XML formatted data files, see “[Setting Up XML Formatted Data Files](#)” on page 178.

To mine data from PDF form fields, you must complete one or both of the following tasks:

1. [“Specifying the Data to Mine from a PDF Form” below](#)
2. [“Mining Multiple Copies of a PDF Form” below](#)

8.3.1 Specifying the Data to Mine from a PDF Form

To specify the data to mine from a PDF XFA form, you must create a PDF form data file. To the engine, the PDF forms defined by PDF form data files are like any other input XML formatted data file. The data file lets you drive the creation and customization of an Exstream Design and Production application using the data you have collected using a PDF form.

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

To specify the data to mine from a PDF form:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select the type of data file you want to create. The option you select is determined by the timing of when you want the data in the PDF form to be available to the engine.
4. From the **File format** list, select **PDF Form**.

5. Click **Finish**.

The data file opens in the Property Panel.

6. Define the data file properties as needed.
7. From the Library, drag the data file to the Edit Panel.
8. Map variables to the form fields you want to mine. You can use the same mapping features that are available for other XML formatted data files.
9. From the **File** menu, select **Save**.

For more information about mapping XML formatted data files, see [“Mapping an XML Formatted Data File” on page 189](#).

8.3.2 Mining Multiple Copies of a PDF Form

If you are mining a single PDF form, or if you are using a connector to access one or more copies of a PDF form, you create and map a PDF form data file as you would create and map

other XML formatted data files. However, there might be times when you must mine data from multiple copies of the same PDF form and you do not want to create a connector. For example, suppose that when a customer returns a PDF form, the form is saved as a separate file on a server. Since the data uses the same PDF form but is not in a single file, you can mine the data but you must separately define the layout of the data and where to find it.

To mine multiple copies of a PDF form, you must complete the following tasks:

1. [“Specifying the Layout of the Data to Mine” below](#)
2. [“Specifying the Location of the PDF Forms to Mine” on the next page](#)

Specifying the Layout of the Data to Mine

To specify the layout of the data you want to mine, you must create a PDF form auxiliary layout file. The auxiliary layout file lets you use the layout of the data elements in the PDF XFA form to associate variables with the tag names and tag values in the form. At run time, the engine reads the data elements in the PDF form and uses the variable mapping to populate variables in an application.

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

To specify the layout of the data to mine:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select **Auxiliary layout**.
4. From the **File format** list, select **PDF Form**.
5. Click **Finish**.

The data file opens in the Property Panel.

6. Click the **Basic** tab.
7. In the **Sample layout** box, select the PDF form used to map the variables.
8. From the Library, drag the auxiliary layout file to the Edit Panel.
9. Map variables to the form fields from which you want to mine customer data. You can use the same mapping features that are available for other XML formatted data files.
10. From the **File** menu, select **Save**.

For more information about mapping XML formatted data files, see [“Mapping an XML Formatted Data File” on page 189](#).

Specifying the Location of the PDF Forms to Mine

After you have defined the layout of the data you want to mine, you must specify the location of the PDF form copies. If you are mining multiple copies, you cannot define the location of the files directly in the input data file. Instead, you must use a placeholder variable. The placeholder variable takes the data from the PDF form copies and passes it through to the engine using the layout you defined in the auxiliary layout file. The engine then populates variables in the application using the data in the PDF form copies. Make sure that the placeholder variable references the auxiliary layout file that defines the data in the PDF form.

For more information about additional placeholder variable properties, see *Importing External Content* in the Exstream Design and Production documentation.

To specify the location of the PDF forms to mine:

1. In Design Manager, in the Library, right-click the **Data Dictionary** heading and select **New Variable**.
The **New Variable** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **Type** list, select **Placeholder**.
4. From the **Placeholder Type** list, select **File map**.
5. If you are using multiple copies of the same form and the variable is an array, select the **Array** check box.
6. Click **Finish**.

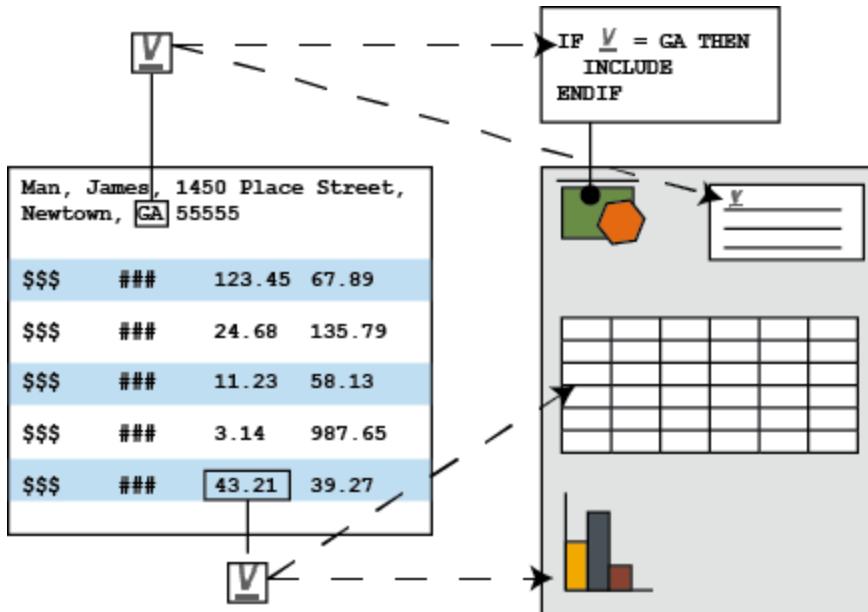
The variable opens in the Property Panel.

7. Go to the **Placeholder** tab.
8. From the **Auxiliary PDF form layout** box, select the auxiliary layout file that defines the data to mine.
9. From the **File** menu, select **Save**.

Chapter 9: Setting Up Data Files to Mine Data from a Print File

If you have created documents on legacy systems or have documents stored in an archive system, you are not required to recreate the data that you used to create those documents. Instead, you can use those documents directly as data. Print files contain information needed to create output on line printers. The Print Miner module lets you use that information in order to leverage the content without making changes to your legacy documents or archival systems. You can mine the data from the archived documents just as if they were flat data files. For example, you can create print files using outside programs, such as an accounting program, and then use them as data file sources for a marketing or financial application in Exstream Design and Production.

Using a print file as a data source



To set up data files to mine data from a print file, you must complete the following tasks:

1. “Creating a Print File Data File” on the next page
2. “Locating Data in a Print File Data File” on page 302
3. “Mapping a Print File Data Area” on page 312

9.1 Creating a Print File Data File

Creating a print file data file, no matter the type, is similar to creating a data file in other formats. First, you create a data file object that defines the data file type and format. Then, you use the data file properties to define the general properties that apply to all of the data in the data file.

Similar to the headers in a delimited data file, print file data files use header pages. Just as with the headers, the engine ignores header pages when reading data.

To create a print file data file, you must complete the following tasks:

1. [“Creating a Print File Data File Object” below](#)
2. [“Defining the Print File Data Format” below](#)

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

9.1.1 Creating a Print File Data File Object

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

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

2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select one of the following options:

- **Customer driver**
- **Reference**

4. In the **File format** list, select **Print file**.
5. Click **Finish**.

The **New Data File** dialog box closes and the print data file opens in the Property Panel for you to define.

9.1.2 Defining the Print File Data Format

After you create the data file object, you must define the format of the data in the print file. Unlike other data file formats, when you select **Print file** from the **File format** list, you must select more options to completely define the file format because print files can have different formats

based on the print provider used to create the document. The options you select determine how the Design Manager and the engine read the data file.

To define the print file data format:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Go to the **Basic** tab.
3. To specify the format of print file, select one the following formatting methods from the **Print data type** list:

| To use this formatting method | Do this |
|---|---|
| Plain text | <ol style="list-style-type: none"> a. From the Print data type list, select Line data. b. In the Records box, enter how many records occur before the page breaks. |
| Line printer file that uses carriage controls. This file format is normally created for legacy printers controlled by print heads. | From the Print data type list, select Line data . |
| AFP line data file that uses carriage controls. This file format is normally created on an IBM system and uses the AFP Page Definition (pagedef) resource to specify where to place data on the page. | From the Print data type list, select AFP line data . |
| Line Conditioned Data Stream (LCDS) file that has Dynamic Job Descriptor Entries (DJDEs). This file format is normally created for a Xerox printer. | Complete the steps in “Defining the IDEN for the Xerox LCDS Format” on page 300 . |
| Fully composed AFP file that Design Manager converts to line data to enable data mining. | Complete the steps “Defining the Conversion Parameters for the Composed AFPDS Format” on page 300 . |

4. To select the type of carriage controls the print file uses, select one of the following options from the **Carriage control** list:
 - **ANSI (wt 0C)**—Design Manager looks for '1' or 'x'0C' to mark the start of a page. This option uses a combination of ASCII and ANSI options.
 - **ASCII**—Design Manager looks for 'x'0C' in defined carriage control locations to define a page. 'x'0C' marks the end of a page.
 - **Machine**—Design Manager looks for 'x'89' and 'x'8B' in the defined carriage control locations to define a page. 'x'8B' marks the start of a page and 'x'89' marks the end of a page. All machine carriage controls are interpreted for data mapping.
 - **ANSI**—Design Manager looks for '1' in the defined carriage control location to define a page. This carriage control is typically located in the first column. '1' marks the start of a page. All ANSI carriage controls are interpreted for data mapping.

5. If the default values defined by the **Carriage control** list are not valid for your print file, for all file formats other than plain text, you can enter custom values. To enter a custom value, do one of the following:

| To | Do this |
|---|--|
| Enter a literal string of characters | <ul style="list-style-type: none"> a. Select the Custom start-of-page characters check box. b. In the adjacent box, enter the characters that identify the start of a page. The limit is 255 characters. Design Manager interprets literal characters you enter as ASCII or EBCDIC based on the encoding of the print stream. |
| Enter a hexadecimal value. Hexadecimal values are often used with machine carriage controls | <ul style="list-style-type: none"> a. Select the Custom start-of-page characters check box. b. In the adjacent box, enter the hexadecimal value. Design Manager uses the numeric value, regardless of the encoding of the print stream. c. Select the Hex check box. |

6. In the **CC offset** box, enter the numbers of columns the carriage controls are offset from the first column. Carriage controls normally appear in the first column, so 0 is the default, meaning that there is no offset.
7. To specify the method in which customer records are separated at the data source, you must do one of the following:

| To | Do this |
|---|---|
| Specify that all customers have the same number of pages | <ul style="list-style-type: none"> a. From the How to identify customers in the data file list, select Each customer has the same number of pages. b. In the Number of pages per customer box, enter the number of pages per customer. |
| Specify that the file uses a single page type to indicate new customers | <ul style="list-style-type: none"> a. From the How to identify customers in the data file drop down list, select A new customer occurs on specified page type(s). b. In the Character offset of the first record-type indicator (0-based) box, specify the beginning location of the first record type indicator. c. In the Size of the indicator box, specify the length of the record type indicator. d. If your data file has sub-indicators, you must do the following: <ul style="list-style-type: none"> i. Select the Multiple indicators check box. The Edit Record Indicators dialog box opens. ii. In the Start box, enter the start location. iii. In the Length box, enter the length of the sub-indicator. iv. Click +. v. Click OK. |

8. Go to the **Advanced** tab.
9. If your print file includes headers that precede a print job and the printer uses to position the printer head, do the following:
 - a. In the **Header** box, enter the number of headers.
 - b. If the print file contains header pages, select the **Pages** check box.
10. From the **Edit** menu, select **Save**.

Defining the IDEN for the Xerox LCDS Format

If your print file uses the Xerox LCDS format, you must define the IDEN. An IDEN is a Job Source Library (JSL) command that specifies the identifier for the Xerox printer command. For example, Exstream Design and Production uses DJDE as the default IDEN when generating Metacode output. When the printer detects DJDE in a specified position within each record, the printer interprets the information on that record as a command, and the information does not appear in the printed document.

To define the IDEN for the Xerox LCDS format:

1. From the **Print data type** list, select **Xerox LCDS**.

The **DJDE IDEN specifications** area appears.
2. In the **Start column** box, enter the starting column for the identification string. Column numbering is zero-based, but the IDEN commands never occur in the first column of the print file. The default value for this field is **1** and represents the second column.
3. In the **Text** box, enter the text used as the IDEN.
4. From the **Character set** list, select one of the following options:
 - **Native**—Design Manager determines the character set based on the platform on which the production environment is running.
 - **ASCII**—A 7-bit character encoding that uses 128 combinations of numbers to represent characters
 - **EBCDIC**—An 8-bit character encoding typically used on IBM mainframes

Defining the Conversion Parameters for the Composed AFPDS Format

The composed AFPDS file format is a print file that is fully composed. Since the file format is not natively line data, Design Manager converts the file to line data to enable data mining. In order to mine the data correctly, you must define the parameters you want Design Manager to use when converting the file.

To define the conversion parameters for the Composed AFPDS format:

- From the **Print data type** list, select **Composed AFPDS**.

The **AFPDS specifications** area appears on the **Basic** tab.

- To specify the dimensions of the smallest character to read, do one of the following:

| To | Do this |
|--|--|
| Manually enter the width and height of the character | <ol style="list-style-type: none"> In the Grid width (pels) box, enter the width of the character. In the Grid height (pels) box, enter the height of the character. |
| Instruct Design Manager to suggest a grid size | <ol style="list-style-type: none"> Click Compute grid. The Compute Grid dialog box opens. In the Enter the number of pages to process box, enter the number of pages you want Design Manager to scan in order to calculate the grid size. The default number of pages is 50. Click OK. You receive a message if Exstream Design and Production encounters any errors. If there are no errors, the Compute Grid Results dialog box opens and shows the Suggested grid width and Suggested grid height. Click OK. The suggested values appear in the Grid width (pels) and Grid height (pels) boxes. |

- If the print file has both portrait and landscape pages, select the **Enlarge line and page size for landscape pages** check box to make sure Design Manager accounts for the longer lines and page widths so no data is lost.
- If your print file might have blank lines, do the following:
 - To keep blank lines that might appear within your page, select the **Keep blank lines within a page** check box.
 - To keep the blank lines that might appear at the bottom of your page, select the **Keep blank lines at the bottom of a page** check box.
- If your print file uses tagged logical elements (TLEs) and no-operation locator search keys (NOP) records at the top of the page, do the following:
 - Select the **Show TLE and NOP records at the top of page** check box.
 - In the **Max length** box, enter the maximum length of the TLE and NOP records. The default is 200.

6. If your AFPDS file uses overlays and formdefs, and they are necessary to determine locations of data within the print file, you must do the following:
 - a. Select the **Use overlays and FORMDEFs** check box.
 - b. In the **FORMDEF name** box, enter the name of the formdef. The maximum length is eight characters. If you enter fewer than eight, Design Manager adds blanks to the name. If you are mapping the file for use on Windows, the formdef file must have an .fdf extension.
 - c. In the **Test directory** box, enter the location of the formdef and overlay files to use when testing. If you are mapping the file for use on Windows, the overlay files must have .ovl extensions.
 - d. In the **Production directory or PDS** box, enter the location of the formdef and overlay files to use when in production. If you are mapping the file for use on Windows, the overlay files must have .ovl extensions.
7. If you do not want specific fonts (for example, Magnetic Ink Character Recognition, also called MICR, fonts) to appear in the Edit Panel when you are mapping the data file, do the following:
 - a. Click  under the **Fonts to ignore** box.

The **Specify the AFP font name to ignore** dialog box opens.
 - b. In the **Specify the name of the AFP font to be ignored** box, enter the name of the font.

If the font name is invalid, you receive a warning message. If the name has fewer than eight characters, Design Manager adds blanks to the name.
 - c. Click **OK**.

The font appears in the **Fonts to ignore** list.

9.2 Locating Data in a Print File Data File

Before you can map data in a print file data file, you must first define where the data can be found. Since a print file includes information that is not true data, it is not like a traditional data file. You must do more than just associate a variable with a data area. You must also define where the data is located within the print file and, rather than simply not mapping variable to a data area, you define areas of the print file data file to ignore.

Tip: If your print file has been formatted to print multiple-up, you can map two copies of the file as separate data files, and map the information from the right page in one file and the information from the left page in another file. You can use this method for any multiple-up, as long as you can define unique spots in the files.

All of the data in a print file data file is relative to another location in the print file. The location can be the upper-left corner of the page, or it can be a location that you define based on content in the print file. To find that location and the data you want to map, you can use the number of lines between the point and the data. If you map your data using line count, the engine uses carriage controls as a locator for mapped data. A carriage control is an instruction in the print file, usually in the first column of the print file, that instructs the print how to print the content. If you do not want the engine to use carriage controls to locate data, you can also use the number of records between the point and the data.

A print file data file is different from other data file formats in that there is no set area for data. Unlike other data files, the same data in a print file data file can appear in a region of the print file and not in an exact position. Where data location in other data file formats is precise, data location in a print file data file can be relative. For example, in the following graphic, the 'Customer_State' variable must appear in a specific delimited data area in the delimited data file, but in the print file data file, it can appear in a larger region.

'Customer_State' variable data area comparison

| | |
|--|--------------------------------|
| G, 3/2/2010, Ab Ovo Usque Ad Mala, 117.47, Sava | Man, James, 1450 Place Street, |
| G, 3/27/2010, Post Festum, 8.94, Savannah, GA,, | Newtown, GA 55555 |
| C, Mildred, James, james M@email.ccm, 1812 East | |
| R, 3/1/2010, Post Cibum, 18.56, Savannah, GA,,, | \$\$\$ ### 123.45 67.89 |
| R, 3/12/2010, Quod Me Nutrit Me Destructit, 35.1 | \$\$\$ ### 24.68 135.79 |
| M, 3/4/2010, Caveat Emptor, 109.62, Savannah, GA | \$\$\$ ### 11.23 58.13 |
| | \$\$\$ ### 3.14 987.65 |
| | \$\$\$ ### 43.21 39.27 |

To locate data in a print file data file, complete the following tasks as needed:

- “Starting a Page Type, Customer, or Section” below
- “Updating an Existing Page Type” on page 306
- “Setting Up a Spot to Define the Relative Location of Data” on page 307

9.2.1 Starting a Page Type, Customer, or Section

Most data files separate data into different records. A print file data file is different because it contains more than just data. Despite the difference, you still must define the way to start

groups of like data. In order to find the data in a print file, you create page types. A page type is similar to a record type indicator in other data file formats. They represent the start of a complete set of data. In some cases, since print files contain content you do not need, page types can also indicate that a page should be ignored when finding data.

Like other data file formats, print file data files can have multiple customers and sections. You use page types to specify where new customers or sections start. After you define the page type, the method in which you define new customers and sections is similar to other data file formats.

To extract data on a page, define a page type, even if the page does not start a new customer or section. When defining page types, define new customer pages first; otherwise, you must change the order in which Design Manager searches the page types. Also define new customer pages before mapping, or make sure that page 1 has the same identifier as all other pages. Design Manager must be able to recognize the identifier for page 1 first so it can distinguish the page from other pages.

To start a new page type, customer, or section:

1. In Design Manager, from the Library, drag the data file to the Property Panel.

The data is shown in yellow, which indicates the page has not been defined.

2. Highlight the first area you find consistently for each customer. For example, you can identify page types by static text such as Page, Dear, Statement, Account Number, or Date.
3. Right-click the highlighted data and select **Page > New Type**.

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

4. In the **Name** box, enter a name for the page.
5. To define how Design Manager locates the page type in the print file, do one of the following:

| To | Do this |
|----------------------------------|---|
| Search using a relative position | <ol style="list-style-type: none">From the Search method list, select Position of page in document.In the Page box, enter the relative position. <p>Note: You cannot use the Position of page in document option to start a customer.</p> |

| To | Do this |
|-------------------------|--|
| Search using exact text | <p>a. From the Search method list, select Text.</p> <p>b. In the Text box, enter the text for which to search.</p> <p>c. To specify the lines, including carriage returns, in which to search for the text, do the following:</p> <ul style="list-style-type: none"> i. In the From box, enter the beginning line. ii. In the To box, enter the ending line. <p>d. To specify the columns in which to search for the text, do the following:</p> <ul style="list-style-type: none"> i. In the From box, enter the beginning column. ii. In the To box, enter the ending column. <p>e. To identify separate lines as different records, select the Search by records check box.</p> |

6. To ignore the content on the page, select the **Ignore** check box. If you select the **Ignore** check box, the page is included in the page count but Design Manager suppresses the engine messages that indicate the page is undefined. If there are not always the same number of header pages in a file, select **Ignore** so Design Manager does not try to read these as customer records.
7. To specify that the page type starts a new customer, select one of the following options from the **Starts customer** list:
 - **None**—A new customer does not start at the selected page type.
 - **This page**—The current page type is the first page type for all customers.
 - **Next page**—The current page type is the last page for all customers.
8. To specify the start of a new section, you must do one of the following:

| To | Do this |
|---|--|
| Not start a new section at the selected page type | From the Starts section list, select None . |
| Always start a new section at the selected page type | <p>a. From the Starts section list, select Always.</p> <p>b. In the Section box, enter the name of the section.</p> |
| Start a new section at the first occurrence of the selected page type | <p>a. From the Starts section list, select First.</p> <p>b. In the Section box, enter the name of the section.</p> |

9. From the **Edit** menu, select **Save**.

9.2.2 Updating an Existing Page Type

After you have created a page type, you can delete it, assign it another type, or update other properties. Reconfiguring the page type properties is helpful if you want to use the same data at different times during processing.

To update an existing page type, complete the following tasks as needed:

- “[Creating Additional Page Types](#)” below
- “[Changing the Search Order of Page Types](#)” below
- “[Setting the Page Type as a Reference](#)” on the next page
- “[Deleting a Page Type](#)” on the next page

Creating Additional Page Types

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Highlight the data area for the page, right-click, and select **Page > New Type**.

A warning message opens.

3. To confirm that you want to create an additional page type, click **Yes**.

You receive a message indicating that the page types might require reordering.

4. Click **OK**.

The message closes and the color of the print file data file updates.

5. From the **Edit** menu, select **Save**.

Changing the Search Order of Page Types

Generally, you are not required to change the default search order. However, when pages with similar matching criteria exist in the same data file, you must adjust the search order. The page search order specifies the order in which Design Manager searches page types in the print file. Design Manager iterates through the page types based on this order until it finds a match.

To change the search order of page types:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Right-click and select **Page > Search Order**.

The **Set Page Search Order** dialog box opens.

3. Select a page name and use and to arrange the pages in the order you want

Design Manager to search them.

4. Click **OK**.

The **Set Page Search Order** dialog box closes.

5. From the **Edit** menu, select **Save**.

Setting the Page Type as a Reference

The page itself can be a reference point if data always appears in the same record and column in relation to the top of the page. When you are using the page as a reference, a spot or text area is not needed to identify the data.

For more information about spot areas, see “[Setting Up a Spot to Define the Relative Location of Data](#)” below.

To set the page type as a reference:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Highlight the page type indicator, right-click, and select **Page > Set as Reference**.
3. From the **Edit** menu, select **Save**.

Deleting a Page Type

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Highlight the page type indicator, right-click, and select **Page > Delete**.
A warning message opens.
3. To confirm that you want to delete the page type, click **Yes**.
The most recently created page type and all references to it are deleted.
4. From the **Edit** menu, select **Save**.

9.2.3 Setting Up a Spot to Define the Relative Location of Data

A spot is an anchor point on a page that lets you map data based on relative positioning to the spot. A spot can occur multiple times in the print file and anchor other spots and data areas. If you define a data area that does not appear in a fixed location on the page, then you must define the relative location of the data using a spot. When you define a spot, you can map data to exact text, an exact position (including a range of lines and columns), or a position relative to a page or information that appears consistently in the file.

To set up a spot to define the relative location of data:

1. In Design Manager, from the Library, drag the data file to the Property Panel.
2. Highlight the area for the spot (up to 255 characters), right-click, and select **Spot > Create**.
3. From the submenu that opens, select one of the following options:
 - **Text Based**—Map the spot to specified text.
 - **Channel Based**—Map the spot to a position based on the value appearing in the channel column.
 - **Spot Based**—Map the spot to a position in relation to another spot.
 - **Any Text in Range**—Map the spot to any text.

The **Reference Spot Properties** dialog box opens.

4. In the **Name** box, enter a unique name for the spot.
5. To specify how spot records operate with data areas that extend for more than one line on a page, select one of the following options from the **Record handling for multiline data areas** list:
 - **Normal**—There are no changes to how the spot operates.
 - **Ignore Records with this Spot**—The record is ignored. For example, use the **Ignore Records with this Spot** option when the record contains a total you want to ignore when you convert multiline data to arrays.
 - **Spot Ends All Data Areas-Exclude**—Ends the processing of all data areas above it, not including current spot
 - **Spot Ends All Data Areas-Include**—Ends the processing of all data areas above it, including current spot
 - **Spot Ends All Data Areas in Sections-Exclude**—Ends the processing of the data area within the current section, not including current spot
 - **Spot Ends All Data Areas in Sections-Include**—Ends the processing of the data area within the current section, including current spot
6. To define how to search for the spot, do one of the following:

| To | Do this |
|--|---|
| Search for a spot using exact text | Complete the steps in “ Searching for a Spot Using Exact Text ” on the next page. |
| Search for a spot relative to another spot | Complete the steps in “ Searching for a Spot Relative to Another Spot ” on page 310 . |
| Search for a spot in a column | Complete the steps in “ Searching for a Spot in a Column ” on page 310 . |

| To | Do this |
|---|--|
| Search for a spot using any text in an area | Complete the steps in "Searching for a Spot Using Any Text in an Area" on page 311 . |
| Search for a spot using a specific occurrence of text | Complete the steps in "Searching for a Spot Using a Specific Occurrence of Text" on page 312 . |

7. To specify how the engine treats section data in the print file, do one of the following:

| If this is true | Do this |
|---|---|
| The spot does not control section data | From the Action list, select None . |
| The spot always starts a new section | <ol style="list-style-type: none">From the Action list, select Start section-always.In the Name box, enter the name of the section. |
| The spot starts a section on its first occurrence | <ol style="list-style-type: none">From the Action list, select Start section-first.In the Name box, enter the name of the section. |
| The engine stops reading information as part of a section and resumes processing on the next page | From the Action list, select Suspend all sections on page . |

8. Click **OK**.

The **Reference Spot Properties** dialog box closes.

9. From the **Edit** menu, select **Save**.

Searching for a Spot Using Exact Text

- From the **Search method** list, select **Text**.
- In the **Text to find or name of spot** box, enter the text for which you want to search.
- To define the area in which the text is found, do the following:
 - To specify the lines, including carriage returns, in which to search for the text, in the **Search range** area do the following:
 - In the **From** box, enter the beginning line.
 - In the **To** box, enter the ending line.
 - To specify the columns in which to search for the text, do the following:

- In the **From** box, enter the beginning column.
 - In the **To** box, enter the ending column.
4. To identify separate lines as different records, select the **Search by records** check box.
5. If the spot occurs multiple times on the region defined by the **Search range** area and you want Design Manager to repeat the data mapping at each occurrence, select the **Multiple** check box. Clear the check box if you want Design Manager to find only the first occurrence of the spot in the range.

Searching for a Spot Relative to Another Spot

Searching for a spot relative to another spot is useful for determining the start of a section. Sometimes unique data that identifies the start of a section occurs before or after the actual start of the section data.

To search for a spot relative to another spot:

1. From the **Search method** list, select **Relative to another spot**.
2. Click  .

The **Select the Spot** dialog box opens.

3. Select a spot defined in the data file.
4. Click **OK**.

The **Select the Spot** dialog box closes and the name of the spot appears in the **Text to find or name of spot** box.

5. If the spot occurs multiple times on the page select the **Multiple** check box.

Searching for a Spot in a Column

1. From the **Search method** list, select **Channel**.
2. To specify the text or spot for which you want to search in the column, do one of the following:

| To | Do this |
|-----------------|--|
| Search for text | In the Text to find or name of spot box, enter the text for which you want to search. |

| To | Do this |
|-------------------------|--|
| Search for another spot | <p>a. Click . The Select the Spot dialog box opens.</p> <p>b. Select a spot defined in the data file.</p> <p>c. Click OK. The Select the Spot dialog box closes and the name of the spot appears in the Text to find or name of spot box.</p> |

3. If the spot occurs multiple times on the page select the **Multiple** check box.
4. To specify that you are searching for an exact occurrence of the text or spot, enter the occurrence in the **Nth occurrence** box.

Searching for a Spot Using Any Text in an Area

1. To specify the location of the data relative to the text, do one of the following:

| To | Do this |
|--|---|
| Use the left-most position of text in the area to calculate the position for data relative to the spot | From the Search method list, select Any text (at text) . |
| Use the left-most position of the area itself to calculate the position for data relative to the spot | From the Search method list, select Any text (left edge of range) . |

2. To define the area in which the text is found, do the following:
 - a. To specify the lines, including carriage returns, in which to search for the text, do the following:
 - In the **From** box, enter the beginning line.
 - In the **To** box, enter the ending line.
 - b. To specify the columns in which to search for the text, do the following:
 - In the **From** box, enter the beginning column.
 - In the **To** box, enter the ending column.
3. To identify separate lines as different records, select the **Search by records** check box.

Searching for a Spot Using a Specific Occurrence of Text

Searching for the character or text at a specified occurrence is useful when the character or text can appear multiple times on a page. Enter the number of the occurrence you want to specify in the **Nth occurrence** box. For example, if Dear occurs three times on the page within the specified range and you want the second occurrence, enter 2 in the **Nth occurrence** box.

To search for a spot using a specific occurrence of text:

1. From the **Search method** list, select **Nth occurrence of text**.
2. In the **Text to find or name of spot** box, enter the text for which you want to search.
3. To define the area in which the text is found, do the following:
 - a. To specify the lines, including carriage returns, in which to search for the text, do the following:
 - In the **From** box, enter the beginning line.
 - In the **To** box, enter the ending line.
 - b. To specify the columns in which to search for the text, do the following:
 - In the **From** box, enter the beginning column.
 - In the **To** box, enter the ending column.
4. To identify separate lines as different records, select the **Search by records** check box.
5. To specify the exact occurrence of the text for which you are searching, enter the occurrence in the **Nth occurrence** box.

9.3 Mapping a Print File Data Area

After you have defined the method in which to find the data, you can map variables to the data areas. The mapping process, after you have defined how to find the data, is similar to mapping other data file formats. In a print file that contains one set of data for each customer, the engine reads all the data for the customer and then builds the customer document. You can have several pages per customer defined the same way. Design Manager searches across all pages for data areas that define each variable.

After you map a data area, Design Manager highlights the data area in a different color. If you want information about what the highlighting colors mean in the Edit Panel, right-click the Edit Panel and select **View > What do colors mean**.

For more information about the data file mapping, see “[Data File Mapping](#)” on page 129.

To map a print file data area, you must complete the following tasks:

1. “[Mapping a Variable to a Data Area](#)” below
2. “[Specifying How the Engine Uses the Data Area During Processing](#)” on the next page
3. “[Specifying How the Engine Searches for the Data Area](#)” on page 316
4. “[Specifying the Input Format of Data in a Data Area](#)” on page 317

9.3.1 Mapping a Variable to a Data Area

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Highlight an unmapped data area that you want to map.
3. To map a variable to the data area, you must do one of the following:

| To | Do this |
|--------------------------|---|
| Map an existing variable | <ol style="list-style-type: none">a. Right-click the area and select Data Area > Map Existing Data Item. The Select Variable dialog box opens.b. Double-click the appropriate variable. The Data Area Properties dialog box opens. |
| Map a new variable | <ol style="list-style-type: none">a. Right-click the area and select Data Area > Map New Data Item. The New Variable dialog box opens.b. In the Name box, enter a name. In the Description box, enter a description (optional).c. From the Type list, select the type of data the variable represents.d. If the variable can contain more than one value, select the Array check box.e. In the Design sample box, enter the text string you want to appear when a designer uses the variable on a page, message, or paragraph in Designer (optional).f. Click Finish. The New Variable dialog box closes and the Data Area Properties dialog box opens. <p>For more information about variable properties, see “Setting Up Variables” on page 59.</p> |

4. Click **OK**.

The **Data Area Properties** dialog box closes.

9.3.2 Specifying How the Engine Uses the Data Area During Processing

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a mapped data area and select **Data Area > Data Area Properties**.
The **Data Area Properties** dialog box opens.
3. To specify what the engine does if the data is not found in the area during processing, select one of the following options from the **Action if data not found** list:
 - **Ignore**—Ignore missing data and continue processing.
 - **Message**—Issue a message and continue processing.
 - **Message and stop**—Issue a message and stop processing.
 - **Message, skip document**—Issue a message and skip the current customer.
4. To specify the amount of data in the data area, do one of following:

| If this is true | Do this |
|--|--|
| Data spans only one line | From the Multiple lines list, select One line only . |
| Data spans a specific number of lines | <ol style="list-style-type: none">a. From the Multiple lines list, select Fixed number of lines.b. In the Lines box, enter the number of lines for the data. |
| Data continues until a blank line or a specific number of lines is reached | <ol style="list-style-type: none">a. From the Multiple lines list, select Until blank line.b. In the Lines box, enter the number of lines for the data. |
| Data continues until specific text or a specified number of lines is reached | <ol style="list-style-type: none">a. From the Multiple lines list, select Until specific text.b. In the Lines box, enter the number of lines for the data.c. In the Spot or text box, enter the text for which you want to search. |

Note: The data does not include the text you specify.

| If this is true | Do this |
|--|--|
| Data continues until a specific spot or a specified number of lines is reached | <p>a. From the Multiple lines list, select Until specific spot.</p> <p>b. In the Lines box, enter the number of lines for the data.</p> <p>c. Click  .</p> <p>The Select the Spot dialog box opens.</p> <p>d. Select a spot defined in the data file.</p> <p>e. Click OK.</p> <p>The Select the Spot dialog box closes and the name of the spot appears in the Spot or text box.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: The data does not include the text in the spot you specify. </div> |
| Data continues until a spot or a specified number of lines is reached | <p>a. From the Multiple lines list, select Until any spot.</p> <p>b. In the Lines box, enter the number of lines for the data.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: The data does not include the text in the spot. </div> |
| Data continues until the next sequential spot is reached | <p>a. From the Multiple lines list, select Until next spot.</p> <p>b. In the Lines box, enter the number of lines for the data.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: The data does not include the text in the spot. </div> |

5. If you select any option other than **One line only** from the **Multiple lines** list, specify how the engine reads data. From the **Data processing method** list, select one of the following options:
 - **Array elements**—Each record in the data area is treated as a separate array element. Blank lines are included for every carriage return. Any lines caused by carriage controls are included as blank elements in the array.
 - **Concatenate**—The data is combined into one continuous string and blank lines are ignored.
 - **Paragraphs**—Each paragraph is treated as a separate array element. This option lets you place and format paragraphs independently in Designer.
 - **Array, no blanks**—Each record in the data area is treated as an array element. Blank lines are ignored whether the lines show in the file or are created by carriage controls.
 - **Array, records only**—Each record in the data area is treated as a separate array element. Blank lines are added only for blank records that appear in the Edit Panel. Blank lines created by carriage controls are ignored.

- **Cells**—Each record in the data area is treated as a separate array element. Underscores, dashes, and blanks are ignored. Use the **Cells** option for data elements that exist in a line data generated table to omit the drawn table lines from the array.
6. If the data area contains confidential information, select **Design sample** from the **Make confidential using** list.
- The data area is marked as confidential but the output does not change at run time unless you use the SECUREOUTPUT engine switch. To produce a new data file that does not contain confidential data, use the COPYINPUT engine switch.
- For more information about the SECUREOUTPUT and COPYINPUT engine switches, see *Switch Reference* in the Exstream Design and Production documentation.
7. Click **OK**.

The **Data Area Properties** dialog box closes.

9.3.3 Specifying How the Engine Searches for the Data Area

1. In Design Manager, from the Library, drag a data file to the Edit Panel.
 2. Right-click a mapped data area and select **Data Area > Data Area Properties**.
- The **Data Area Properties** dialog box opens.
3. To select the method the engine uses to search for the data area, do one of the following:

| To | Do this |
|--|---|
| Search for the data area using the vertical and horizontal offsets based on the top left of the page | From the Search method list, select Relative to page . |
| Search for the data area based on the location of a spot | <ol style="list-style-type: none">a. From the Search method list, select Relative to active spot.b. Click . The Select the Spot dialog box opens.c. Select a spot defined in the data file.d. Click OK. <p>The Select the Spot dialog box closes and the name of the spot appears in the Spot box.</p> |

4. To define the vertical offset, do the following:

- a. In the **Vertical offset** box, enter the vertical location of the data area. Design Manager reads the vertical offset starting with the first line mapped, and counting down to the next line mapped.
- b. If the vertical offset is measured using records, select the **Records** check box.
5. In the **Horizontal offset** box, enter the horizontal location of the data area.
6. If the data area is defined in a line and you are mapping data that overprints the line, select the **On overprint** check box. Only one overprint can be used per line. Overprint is valid only when using lines, since records are the physical count and lines interpret the 0, +, - characters.
7. Click **OK**.

The **Data Area Properties** dialog box closes.

9.3.4 Specifying the Input Format of Data in a Data Area

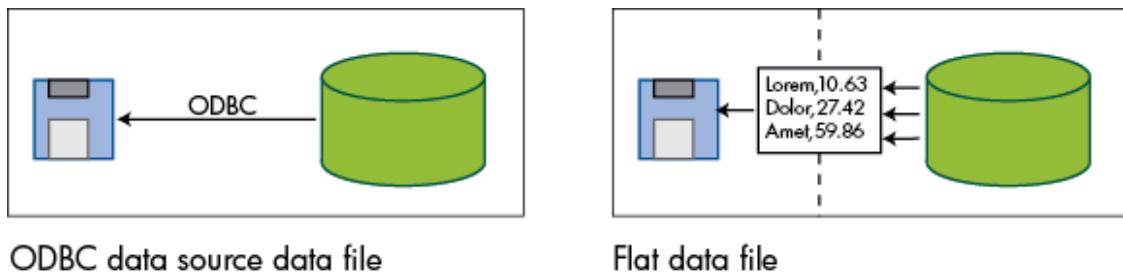
1. In Design Manager, from the Library, drag a data file to the Edit Panel.
2. Right-click a mapped data area and select **Data Area > Data Area Properties**.
The **Data Area Properties** dialog box opens.
3. To specify the format of the data area, select an option from the **Format** list. The options vary based on the variable type mapped to the data area.
For more information about the available format options, see the “[Data Area Input Formats](#)” on page 162.
4. If you are mapping a currency or a floating variable and you want to specify the location of the decimal when it does not appear in the data, enter the number of implied digits in the **Implied digits** box. Enter a positive number (for example, 1 or 5) to move the decimal to the left. Enter a negative number (for example, -1 or -5) to move the decimal to the right. You can have a maximum of nine implied digits.
5. In the **Length** box, enter the width of the highlighted area in the data file.
6. To stop data area at a particular string of text, enter a string of text in the **End text** box. The string of text you enter is not included in the data area.
7. Click **OK**.

The **Data Area Properties** dialog box closes.

Chapter 10: Setting Up Data Files to Extract Data from an ODBC Data Source

With most data file formats, in order to access and map data, you must first extract data from a data source and create a data file. The ODBC Access module, however, removes the data extraction step. It lets you use Open Database Connectivity (ODBC) to directly access data in a data source. With the ODBC Access module, you can access and map variables to the contents of ODBC relational databases without building custom programs to extract the data or mapping variables to flat files. Instead, you create a data file object and use the built-in tools to select a stored procedure or to create a Structured Query Language (SQL) query that you can use to select database structures you want to map.

ODBC data source and flat data file comparison



By using ODBC to access a data source directly instead of through a data file, you can help ensure that the engine accesses the latest information available when you package an application and produce output. A database can be continuously updated; a fixed, flat data file source cannot. ODBC data queries can be slower than flat file data sources, though, due to the performance of the database and the network. You can also use a centralized source for an application's data instead of a number of separate data file sources for customer driver, initialization, and reference files. Its client-server structure gives you the flexibility to connect to data sources either on your own workstation or on remote servers.

Setting up an ODBC data source data file is like setting up other data file formats. You follow the same general procedure. Keep in mind, however, that you cannot use section data with ODBC data source data files. In addition, the text data type is not supported in ODBC data source data files for PostgreSQL. You must use the varchar data type with no size specification instead.

This chapter discusses the following topics:

- “[Supported ODBC Drivers](#)” on the next page
- “[Connecting to an ODBC Data Source](#)” on page 321

- “[Selecting the Data Source Structure to Map](#)” on page 324
- “[Mapping an ODBC Data Source Data File](#)” on page 342

10.1 Supported ODBC Drivers

In order to access the data in a database, you must use a Data Source Name (DSN). A DSN lets you connect to the database without having to navigate to the database each time. To create a DSN, you use an ODBC driver. The ODBC driver you use varies based on your database platform and production environment platform. The driver you use also depends on the type of production engine you are using: 32-bit or 64-bit. In addition, keep in mind that ODBC drivers from various vendors vary in feature capabilities and SQL support. Some ODBC Access module capabilities are not available with certain drivers.

For more information on the 32-bit and 64-bit versions of the Exstream Design and Production production engines, see *Preparing Applications for Production* in the Exstream Design and Production documentation.

This section discusses the following topics:

- “[Supported Windows ODBC Drivers](#)” below
- “[Supported UNIX/Linux ODBC Drivers](#)” on the next page
- “[Supported z/OS ODBC Drivers](#)” on page 321

10.1.1 Supported Windows ODBC Drivers

The following table lists the ODBC drivers that are supported on Windows for each database type that Exstream Design and Production supports.

Supported Windows ODBC Drivers

| Database type | Supported ODBC drivers |
|--|---|
| DB2 (Server running on Linux/Unix/Windows) | IBM Data Server Client or IBM DB2 Connect ODBC driver |
| DB2 (Server running on z/OS) | DataDirect Connect for ODBC DB2 Wire Protocol Driver (DBCS DB2 on z/OS) |
| Oracle | Oracle Client ODBC Driver |
| SQL Server | Microsoft SQL Server ODBC Driver |
| PostgreSQL | PostgreSQL ODBC Driver (psqlODBC) |

10.1.2 Supported UNIX/Linux ODBC Drivers

The following table lists the ODBC drivers that are supported on UNIX and Linux platforms for each database type that Exstream Design and Production supports.

Supported UNIX/Linux ODBC drivers

| Database type | Operating system | Supported ODBC drivers |
|---------------|------------------|---|
| DB2 | AIX | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS and DBCS engines • unixODBC with DB2 Client on 32-bit and 64-bit for SBCS and DBCS engines |
| | HP-UX Itanium | DataDirect Connect for ODBC on 64-bit for SBCS engines only |
| | Solaris | DataDirect Connect for ODBC on 32-bit SBCS and 64-bit DBCS engines |
| | Linux | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS engines only • unixODBC with DB2 Client on 32-bit and 64-bit for SBCS engines only |
| Oracle | AIX | DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS and DBCS engines |
| | HP-UX Itanium | DataDirect Connect for ODBC on 64-bit for SBCS engines only |
| | Solaris | DataDirect Connect for ODBC on 32-bit SBCS and 64-bit DBCS engines |
| | Linux | DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS only |
| SQL Server | AIX | DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS and DBCS engines |
| | HP-UX Itanium | DataDirect Connect for ODBC on 64-bit for SBCS only |
| | Solaris | DataDirect Connect for ODBC on 32-bit SBCS and 64-bit DBCS engines |
| | Linux | DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS only |

Supported UNIX/Linux ODBC drivers, continued

| Database type | Operating system | Supported ODBC drivers |
|---------------|------------------|--|
| PostgreSQL | AIX | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS and DBCS engines • PostgreSQL ODBC Driver (psqlODBC) on 32-bit and 64-bit for SBCS and DBCS engines |
| | HP-UX Itanium | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 64-bit for SBCS engines only • PostgreSQL ODBC Driver (psqlODBC) on 64-bit for SBCS engines only |
| | Solaris | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 32-bit SBCS and 64-bit DBCS engines • PostgreSQL ODBC Driver (psqlODBC) on 32-bit SBCS engines only |
| | Linux | <ul style="list-style-type: none"> • DataDirect Connect for ODBC on 32-bit and 64-bit for SBCS engines only • PostgreSQL ODBC Driver (psqlODBC) on 32-bit and 64-bit for SBCS engines only |

10.1.3 Supported z/OS ODBC Drivers

On z/OS, the supported ODBC driver is the IBM DB2 CLI Driver, which is available for the 32-bit and the 64-bit versions of the Exstream engine.

Important: The engine supports only reading from SBCS ODBC data files.
DBCS ODBCS data files and writing to SBCS OBCS data files are not supported.

10.2 Connecting to an ODBC Data Source

Creating an ODBC data source data file, no matter the type, is similar to creating a data file in other formats. First, you create a data file object that defines the data file type and format. Then, you use the data file properties to define the general properties that apply to all of the data in the data file.

As with other data file formats, you use a data file to define and point to existing data sources created and stored outside of Exstream. However, unlike other data file formats, ODBC data source data files do not use file paths to find the data sources. Instead, since the data is not extracted, you use the data file to select a DSN that connects to the database.

To connect to an ODBC data source, you must complete the following tasks:

1. [“Creating an ODBC Data Source Data File Object” below](#)
2. [“Defining the Location of the Data Source For Testing and Production” below](#)

For more information about additional data file properties, see the section in [“Setting Up Data Files” on page 9](#) that corresponds to the data file type you are defining.

10.2.1 Creating an ODBC Data Source Data File Object

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.

The **New Data File** dialog box opens.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. From the **File type** list, select one of the following options:
 - **Auxiliary layout file**
 - **Customer driver file**
 - **Initialization file**
 - **Post-sort initialization file**
 - **Post-sort report file**
 - **Reference file**
 - **Report file**
4. From the **File format** list, select **ODBC data source**.
5. Click **Finish**.

The **New Data File** dialog box closes and the ODBC data source data file opens in the Property Panel for you to define.

10.2.2 Defining the Location of the Data Source For Testing and Production

After you create an ODBC data source data file object, you must set additional properties so the engine can access the data source that you want to use. You set the properties of an ODBC data source data file like a traditional data file, including how to locate the data source during testing and production. Just as with other data file formats, you define the location of the data

source for testing and production separately. For example, to make testing faster, you might use a smaller database, such as Microsoft SQL Server Express, with sample data for testing, but a larger, complete Oracle database for production.

However, unlike other data file formats, when you use an ODBC data source data file, you do not enter a file path to define the location of the data source. Instead, you select a DSN and a schema that points to the data source. A DSN lets you select a symbolic name to connect to the database without having to navigate to the database each time. A schema is a collection of data tables that contain similar data.

To define the location of the data source for testing and production:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Property Panel.
2. To define the location of the test data source, do the following:
 - a. Go to the **Test Data Source** tab.
 - b. In the **DSN to use in test mode** box, select the DSN to use to the test data source. The options vary based on the DSNs available on your workstation.
 - c. If the database is protected, do the following:
 - In the **Username** box, enter the user name.
 - In the **Password** box, enter the password.
 - d. If you are using an Oracle, a SQL Server, or a DB2 data source, from the **Schema** list, specify the schema or owner of the tables you want to use within the DBMS. If you do not select an option from the **Schema** list, the schema will default to the database user name.
3. To define the location of the production data source, do the following:
 - a. Go to the **Production Data Source** tab.
 - b. In the **DSN to use in production** box, select the DSN to use to the production data source.
 - c. If the database is protected, do the following:
 - In the **Username** box, enter the user name.
 - In the **Password** box, enter the password.
 - d. Enter the schema for the final production in the **Schema** box. By default, the **Schema** box is populated using the schema selected from the **Schema** list on the **Test Data Source** tab.
 - e. Click the **Test connection** button to verify the data file can connect to the data source for production.

You receive a message informing you whether the connection attempt was successful.

4. From the **Edit** menu, select **Save**.

10.3 Selecting the Data Source Structure to Map

After you create the data file object, but before you map the data, you must select the data you want to map. In other data file formats, the data you map is defined when the data is extracted from the database. With ODBC data source data file, however, you define the data as part of the data file set up process. You are not required to map the entire data source. Instead, for each record, you can filter the data extracted to include only specific schemas, tables, and columns.

To select the data source structure to map, complete the following tasks as needed:

- [“Using the Table Query Method to Select Data” below](#)
- [“Using the Stored Procedure Method to Select Data” on page 338](#)

10.3.1 Using the Table Query Method to Select Data

The table query method lets you use the built-in tools to create a SQL query to select the tables and columns you want to map in each record. The query lets you describe the data you want to retrieve and map. The table query method is manual and can take longer than the stored procedure method, but it gives you more control over the tables, columns, and other options. You can also use the tools to filter, combine, and organize the tables and columns to retrieve just the data you need in the order you need. In addition, some database platforms do not support stored procedures.

To use the table query method to select data, you must complete the following tasks:

1. [“Defining the Record Properties” on the next page](#)
2. [“Selecting the Tables to Include in the Query” on page 326](#)
3. [“Selecting the Columns to Include in the Query” on page 327](#)
4. [“Joining Tables with Shared Columns in the Query” on page 330](#)

You can also complete the following optional tasks as needed:

- “[Filtering Data from the Query Results](#)” on page 333
- “[Sorting the Query Results](#)” on page 334
- “[Editing the SQL Query Manually](#)” on page 336

Defining the Record Properties

Similar to defining records in flat data files, you define each record in an ODBC data source data file separately. You define properties such as the start of a customer. However, since the data has not been extracted from the data source, you query each record individually. You configure properties such as which rows to include and how many.

When you define a record, unless every customer has the same number of records, you must identify where a new customer starts.

Caution: If you do not identify the start of a customer, the engine does not reset variables to have zero elements, which can cause incorrect results at run time if you want to reset variables at the customer level.

To define the record properties:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. To create the record do one of the following:

| To | Do this |
|---------------------------|---|
| Create the first record | If no records have been added to the data file, the ODBC Record Properties dialog box opens automatically when you drag the data file to the Edit Panel. |
| Select an existing record | Right-click the Edit Panel and select Record > Record Properties . The ODBC Record Properties dialog box opens. |
| Create a new record | Right-click the Edit Panel and select Record > Add Record . The ODBC Record Properties dialog box opens. |

3. From the **Record source** list, select **Table query**.
4. In the **Rows to display** box, enter the number of rows from each column you want to appear in the Edit Panel. The default is 1.
5. To return only unique row values, select the **Select distinct rows** check box. If you clear the **Select distinct rows** check box, the query returns all qualifying row values, including duplicates.
6. If you are mapping a customer driver file and the record is the start a new customer, select the **Start new customer** check box.

7. Configure the remaining properties.
8. Click **OK**.
9. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

Selecting the Tables to Include in the Query

The first step in defining the data you want to map is to select the tables you want to include in the query. The tables available vary based on the schema that you selected on the **Test Data Source** tab in the data file object properties. A table is the structure that stores the data in the data source. It is made up of columns, which define data type, and rows, which contain the data values.

When you select the tables to include in the query, you are adding FROM statements to the SQL query. For example, the sample below is sending a query to the Contacts and Employee tables.

Sample table SQL query

```
SELECT [schema].Contacts.ContactsID,  
       [schema].Employees.EmployeeID,  
       [schema].Contacts.FirstName,  
       [schema].Contacts.LastName,  
       [schema].Employees.DepartmentName,  
       [schema].Employees.FirstName,  
       [schema].Employees.LastName,  
       [schema].Contacts.WorkPhone,  
       [schema].Employees.EmailName  
① FROM [schema].Contacts,  
      [schema].Employees  
 WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID  
   AND [schema].Contacts.ContactsID = <Contacts_EmployeeID>  
 ORDER BY [schema].Contacts.ContactsID ASC
```

| | |
|---|-----------------|
| 1 | Tables to query |
|---|-----------------|

For more information about defining the test data source schema, see “[Defining the Location of the Data Source For Testing and Production](#)” on page 322.

To select the tables to include in the query:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Go to the **Tables** tab.

4. To select the table or tables you want to map, do one or more of the following:

| To | Do this |
|---|---|
| Select a table | <ol style="list-style-type: none">In the Available list, select the table you want to add.Click Add. <p>The table is added to the Selected list.</p> <p>Tip: To select multiple tables, press and hold the CTRL key while selecting tables. To select a list of consecutive tables, press and hold the SHIFT key while selecting the first and last table you want in the list.</p> |
| Change the order of the tables in the query | <ol style="list-style-type: none">In the Selected list, select the table you want to reorder.Click  to move the table up in the order or click  to move the table down. <p>Note: The order of the tables in the query dictates the order the data appears on the other tabs (Columns, Join, Filter, Order).</p> |
| Remove a table | <ol style="list-style-type: none">In the Selected list, select the table you want to remove.Click Remove. |

5. If more than one table is in the **Selected** list and you want enter a different name in order to simplify reading the code in the **SQL** tab, do the following:
- In the **Selected** list, select the table.
 - Enter a name in the **Alias** box. Aliases can help you avoid ambiguity when you join a table to itself. Aliases appear only in Exstream Design and Production. The original names of tables still appear in the data source.
6. Click **OK**.
7. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

Selecting the Columns to Include in the Query

After you select tables, the next step in defining the data you want to map is to select the columns you want to include in the query. Data is retrieved from the columns you select. The available columns vary based on the tables you selected on the **Tables** tab.

The **Available** list includes columns from all of the selected tables. Each column must have a unique name. If you selected multiple tables on the **Tables** tab, the table name and a period precede each column name (for example, Account.OpeningBalance). Binary Large Object (BLOB) columns do not appear in the **Available** list. By default, primary keys automatically appear in the **Selected** list. If supported by the database, foreign keys also appear in the **Selected** list.

When you select the columns to include in the query, you are adding SELECT statements to the SQL query. For example, in the example below Contacts.ContactsID and Employees.EmployeeID are the primary keys and the other columns are columns selected in the **Columns** tab.

Sample column SQL query

```

SELECT [schema].Contacts.ContactsID,
       [schema].Employees.EmployeeID,
       [schema].Contacts.FirstName,
       [schema].Contacts.LastName,
       [schema].Employees.DepartmentName,
       [schema].Employees.FirstName,
       [schema].Employees.LastName,
       [schema].Contacts.WorkPhone,
       [schema].Employees.EmailName
  FROM [schema].Contacts,
       [schema].Employees
 WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID
   AND [schema].Contacts.ContactsID = <Contacts_EmployeeID>
 ORDER BY [schema].Contacts.ContactsID ASC

```

| | |
|----------|------------------|
| 1 | Tables to query |
| 2 | Primary keys |
| 3 | Columns to query |

To select the columns to include in the query:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Make sure that tables were selected on the **Tables** tab.
4. Go to the **Columns** tab.
5. To select the column or columns you want to map, do one or more of the following:

| To | Do this |
|--|--|
| Select a single column | <p>a. In the Available list, select the column you want to add.</p> <p>b. Click Add.</p> <p>The column is added to the Selected list.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Tip: To select multiple columns, press and hold the CTRL key while selecting columns. To select a list of consecutive columns, press and hold the SHIFT key while selecting the first and last column you want in the list. </div> |
| Select all columns | <p>a. Click Add All.</p> <p>The ODBC Column Mapping Options dialog box opens.</p> <p>b. In the Binary column mapping area, select one of the following radio buttons:</p> <ul style="list-style-type: none"> • Map columns as binary value—The data is not viewable in a text editor. • Map columns as text—The data is viewable in a text editor. Text columns are automatically set to Map columns as text. <p>c. In the Maximum width for text columns box, enter the number of characters in the widest text column. The default is 200. The value of the largest text column must be less than 32K.</p> <p>d. Click OK.</p> |
| Change the order columns appear in the query | <p>a. In the Selected list, select the column you want to reorder.</p> <p>b. Click  to move the column up in the order or click  to move the column down.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> Note: This also changes the order of the columns in the Edit Panel. </div> |
| Remove a column | <p>a. In the Selected list, select the column you want to remove.</p> <p>b. Do one of the following:</p> <ul style="list-style-type: none"> • To remove a single column, click Remove. • To remove all of the selected columns, click Remove All. The key columns are not removed. |

6. If you want to adjust the mapping options for a column, do the following:
 - a. Select a column in the **Selected** list.
 - b. In the **Mapping Options** area, select one of the following radio buttons:
 - **Text**—The data is viewable in a text editor. Text columns are automatically set to **Text**.
 - **Binary**—The data is not viewable in a text editor.

- c. If you are defining a string (text) column, in the **Text width** box, enter the number of characters the widest text column can be.

The value of the largest text column must be less than 32K.

- d. In the **Null value** box, enter the value to appear if the value in the column is unknown or does not match the type defined by the column properties.

7. Click **OK**.

8. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box; otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

Joining Tables with Shared Columns in the Query

If you have selected tables that share one or columns, you can combine the tables using values common to each table. The **Join** tab is not available for auxiliary layout, report, and post-sort report files.

When you join tables, you are adding either a **WHERE** statement or a **JOIN** statement to the SQL query. The type of statement you add varies based on the type of join you are completing.

Sample join SQL query

```

SELECT [schema].Contacts.ContactsID,
       [schema].Employees.EmployeeID,
       [schema].Contacts.FirstName,
       [schema].Contacts.LastName,
       [schema].Employees.DepartmentName,
       [schema].Employees.FirstName,
       [schema].Employees.LastName,
       [schema].Contacts.WorkPhone,
       [schema].Employees.EmailName
  FROM [schema].Contacts,
       [schema].Employees
 WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID
   AND [schema].Contacts.ContactsID = <Contacts_EmployeeID>
 ORDER BY [schema].Contacts.ContactsID ASC

```

| | |
|---|-----------------|
| 1 | Tables to query |
| 2 | Primary keys |

| | |
|---|------------------|
| 3 | Columns to query |
| 4 | Join method |

To join tables with shared columns in the query:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Make sure that tables were selected on the **Tables** tab.
4. Go to the **Join** tab.
5. From the **Table** and **Column** lists, select the tables and columns you want to join. You can select any of the columns for the tables that you selected in the **Tables** tab, regardless of whether they are selected in the **Columns** tab.
6. To add additional joins, select the check box at the beginning of the next row and repeat step 2 through step 5. You can complete up to five rows, depending on the amount of joins you want.
7. To select a row retrieval method for the engine, select one of the following options from the **Join type** list:

| Option | Description | SQL statement |
|-------------------------|--|---------------|
| Old style | Retrieve all rows in both tables. | WHERE |
| Inner Join | Retrieve only those rows that match in both tables. Inner joins create a new result table by combining column values of two tables based upon the join-predicate. The query compares each row of the first (left) table with each row of the second (right) table to find all pairs of rows which satisfy the join-predicate. | JOIN |
| Left Outer Join | Retrieve both matching and non-matching rows from the table on the left side of the join. The result of a left outer join always contains all records of the left table, even if the join-condition does not find any matching records in the right table. A left outer join returns all the values from the left table, plus matched values from the second table (or NULL in case of no matching join predicate). If the left table returns one row and the right table returns more than one matching row for it, the values in the left table are repeated for each distinct row on the right table. | LEFT JOIN |
| Right Outer Join | Retrieve both matching and non-matching rows from the table on the right side of the join. The result of a right outer join always contains all records of the right table, even if the join-condition does not find any matching records in the left table. A right outer join returns all the values from the right table and matched values from the left table (NULL in case of no matching join predicate). If the right table returns one row and the left table returns more than one matching row for it, the values in the right table are repeated for each distinct row on the left table. | RIGHT JOIN |

| Option | Description | SQL statement |
|------------------------|---|---------------|
| Full Outer Join | Retrieve both matching and non-matching rows from both tables. The joined table contains all records from both tables, and fills in NULLs for missing matches on either side. | FULL JOIN |
| Cross Join | Use a Cartesian cross query to match each record in the left table to a record in the right table. The result of a cross joins includes all records which satisfy the join predicate. | CROSS JOIN |

8. Click **OK**.
9. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

ODBC Driver Support for Join Options

Depending on the ODBC driver you use, some join options might be unavailable. The following table shows current driver support for the various types of join options in Exstream Design and Production. Verify that your ODBC driver meets the criteria for Design Manager options on the **Join type** list.

ODBC driver support for join type

| Driver | Supported join type |
|--------------------|---|
| SQL ODBC Driver | <ul style="list-style-type: none"> • Old style • Inner Join • Left Outer Join • Right Outer Join • Full Outer Join • Cross Join |
| Oracle ODBC Driver | <ul style="list-style-type: none"> • Old style • Inner Join • Left Outer Join • Right Outer Join • Full Outer Join |

ODBC driver support for join type, continued

| Driver | Supported join type |
|---------------------|---|
| IBM DB2 ODBC Driver | <ul style="list-style-type: none">• Old style• Inner Join• Left Outer Join• Right Outer Join• Full Outer Join |

Filtering Data from the Query Results

You might not need all of the data that could be returned by the query. You can increase engine productivity by filtering unneeded data from the results. You can filter results based on a variable or a static value. Any rows that do not meet your filter criteria are not returned in the results. The **Filter** tab is not available for auxiliary layout, report, and post-sort report files. When you filter data, you are adding a WHERE statement to the SQL query.

Sample filter SQL query

```
SELECT [schema].Contacts.ContactsID,  
       [schema].Employees.EmployeeID,  
       [schema].Contacts.FirstName,  
       [schema].Contacts.LastName,  
       [schema].Employees.DepartmentName,  
       [schema].Employees.FirstName,  
       [schema].Employees.LastName,  
       [schema].Contacts.WorkPhone,  
       [schema].Employees.EmailName  
  FROM [schema].Contacts,  
        [schema].Employees  
 WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID  
   AND [schema].Contacts.ContactsID = <Contacts EmployeeID>  
 ORDER BY [schema].Contacts.ContactsID ASC
```

The diagram illustrates the components of the SQL query with numbered callouts:

- 1**: Points to the `FROM [schema].Contacts,` and `[schema].Employees` clauses.
- 2**: Points to the `SELECT [schema].Contacts.ContactsID,` through `[schema].Employees.EmailName` clauses.
- 3**: Points to the `WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID` clause.
- 4**: Points to the `AND` keyword.
- 5**: Points to the `<Contacts EmployeeID>` placeholder in the WHERE clause.

| | |
|----------|------------------|
| 1 | Tables to query |
| 2 | Primary keys |
| 3 | Columns to query |
| 4 | Join method |
| 5 | Filter condition |

To filter data from the query results:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Make sure that tables were selected on the **Tables** tab.
4. Go to the **Filter** tab.
5. From the **Table** list, select the table you want to filter.
6. From the **Column** list, select the column you want to filter.
7. From the **Condition** list, select the condition you want to use when comparing record values.
8. In the **Compare To** box, do one of the following:
 - Select the variable that contains the values you want to compare to the variable in the **Variable** box.
 - Enter a text string surrounded by double quotation marks. Anything not in quotation marks is considered the name of a variable. Always add double quotation marks on either side of a string, amount, date, or number.
9. If you require more than one filter, select the check box on the next row, and select one of the following:
 - **and**—Both of the adjoining condition statements must be true in order for the rule to be included/excluded from the customer output.
 - **or**—Any of the adjoining condition statements can be true in order for the rule to be included/excluded from the customer output.
10. Click **OK**.
11. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

Sorting the Query Results

Normally, the query results are in primary key order, but the order is not guaranteed, particularly when you use filters. If your application requires the results in a certain order, you can use the columns you selected to sort the results either in ascending or descending order. The results are sorted by the values in the columns you select. Specifying the order lets you reduce database query time and establish the sequence of information in the composed output. If you leave the default options on each tab of the **ODBC Record Properties** dialog box, the engine reads data in ascending order. The **Order** tab is not available for auxiliary layout, report, and post-sort report files.

When you order the results, you are adding an ORDER BY statement to the SQL query.

Sample Order SQL query

```

SELECT [schema].Contacts.ContactsID,
       [schema].Employees.EmployeeID,
       [schema].Contacts.FirstName,
       [schema].Contacts.LastName,
       [schema].Employees.DepartmentName,
       [schema].Employees.FirstName,
       [schema].Employees.LastName,
       [schema].Contacts.WorkPhone,
       [schema].Employees.EmailName
  FROM [schema].Contacts,
       [schema].Employees
 WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID
   AND [schema].Contacts.ContactsID = <Contacts EmployeeID>
 ORDER BY [schema].Contacts.ContactsID ASC

```

| | |
|---|------------------|
| 1 | Tables to query |
| 2 | Primary keys |
| 3 | Columns to query |
| 4 | Join method |
| 5 | Filter condition |
| 6 | Order |

To sort the query results:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Make sure that tables were selected on the **Tables** tab.
4. Go to the **Order** tab.
5. Select the check box beside the order command you want to create.
6. From the **Table** list, select the table you want to order.
7. From the **Column** list, select the column you want to order.

8. From the **Order** list, select the order in which to sort the results:
 - **Ascending**
 - **Descending**
9. If you want to order another column, select the check box on the next row.
10. Click **OK**.
11. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see ["Mapping an ODBC Data Source Data File" on page 342](#).

Editing the SQL Query Manually

If the SQL query is not as you expect, you can use the **SQL** tab to edit the view and edit the Microsoft SQL query the engine sends to the database in design and production mode runs. For example, if, during a test, you do not receive the results you expect you can edit the default query. You can also edit the query if you need more join, filter, or order statements than can be created using the options on the **ODBC Record Properties** dialog box. You should be an expert user and experienced in SQL coding to edit the SQL statement.

When you manually edit a complex query, keep in mind the following considerations:

- Any variable you select on the **Filter** tab appears in the WHERE statement in brackets. For example, if you select the 'Contacts_EmployeeID' variable, it appears as <Contacts_EmployeeID> in the query.
- If you want to select a variable on the **SQL** tab, represent the variable with a question mark (?). You can then select the variable on the tab. If you define one variable on the **SQL** tab, you must define all of the variables in the query on the **SQL** tab. The sequence of the variables you select must match the sequence of the question marks in the query.

Important: If you remove or add a question mark during editing, the variables listed to the right of the editing box do not adjust automatically; you must manually reorder them.

- Exstream Design and Production uses placeholders in the place of schemas in SQL queries in the format of [schema]. At run time, the engine either replaces the [schema] placeholder with the actual schema, or removes it if you do not specify a schema. If you manually edit the SQL query, use the [schema] placeholder ("schema" must be lowercase).

For example, if you manually edit the WHERE clause, and want to find every record where the LAST_NAME in the CUSTOMER table is equal to Adams, enter WHERE [schema].CUSTOMER.LAST_NAME = 'Adams'.

Note: If you are setting a condition equal to a character string (as opposed to an integer, for example), the string must be enclosed in single quotations (').

- Once you manually edit the SQL statement, the **Join**, **Filter**, and **Order** tabs are removed and cannot be reinstated. Any future changes to the query must be made manually on the **SQL** tab.

To edit the SQL statement manually:

- In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
- Right-click the Edit Panel and select **Record > Record Properties**.
- Go to the **SQL** tab.
- Select the **Manually edit complex query (SELECT and FROM clause must remain unchanged)** check box.

Note: If you make edits to the SQL query without checking this box, the edits will not be saved.

- Edit the query.
- If you are entering a variable, do the following:
 - Enter ? to use the value of a variable as a parameter.

Sample Variable Edit - Before

```
WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID  
      AND [schema].Contacts.ContactsID = <Contacts EmployeeID>  
ORDER BY [schema].Contacts.ContactsID ASC
```

Sample Variable Edit - After

```
WHERE [schema].Contacts.ContactsID = [schema].Employees.EmployeeID  
      AND [schema].Contacts.ContactsID = ?  
ORDER BY [schema].Contacts.ContactsID ASC
```

A variable box becomes active on the right-hand side of the dialog box.

- Select the variable that will replace the ?.



The first variable box always corresponds to the first ? in the statement, the second variable box always refers to the second ? in the statement, and so forth. To add more than eight variables, click to access additional variable boxes.

7. Click **OK**.
8. If you are creating new variables to map to the data areas, complete the properties on the **Map ODBC Columns to Variables** dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see [“Mapping an ODBC Data Source Data File” on page 342](#).

10.3.2 Using the Stored Procedure Method to Select Data

The stored procedure method lets you automatically select columns, tables, and preferences. Stored procedure objects (sometimes referred to as stored procs) reside in a database and contain predefined SQL, local variables, and built-in functions to reduce the time and effort to complete database tasks. They use input parameters to select the data to include and output parameters to return the data.

The stored procedure method streamlines database mapping and is faster than the table query method, but it does not let you control table, column, and preference options. With stored procedures, you can view the SQL statement the engine sends to the database in design and production mode runs, but you cannot manually edit the SQL query in a stored procedure. Some database platforms do not support stored procedures. Currently, Exstream Design and Production supports stored procedures with SQL Server and IBM DB2 only. SQL Server supports return values; however, DB2 does not.

Stored procedures return a recordset which you can use to map an ODBC data source data file in the Edit Panel. You can use stored procedures with customer driver files, reference files, and initialization files only.

To use the stored procedure method to select data, complete the following tasks as needed:

- “[Setting Up an ODBC Data Source Data File Object for a Stored Procedure](#)” below
- “[Defining the Record Properties](#)” below
- “[Defining the Input Parameters for the Stored Procedure](#)” on the next page

Setting Up an ODBC Data Source Data File Object for a Stored Procedure

If you want to use a stored procedure to create and map an ODBC data source data file, you must select and define the stored procedure to use to create records.

You can use the value of the variable you select in the **Value to pass to stored procedure** box when creating the primary key in the table. After the stored procedure runs, the variable will also contain the value of the output parameter from the procedure. Refer to the stored procedure to determine the actual information that the variable must pass to the database. The **Value to pass to stored procedure** box is not available with auxiliary layout files.

To set up an ODBC data source data file object for a stored procedure:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Property Panel.
2. Go to the **Advanced** tab.
3. In the **Record creation stored procedure** box, enter the name of the stored procedure that generates unique object IDs for records you write to the database. These IDs are typically long integers.
4. If you use multiple numbering schemas, enter the variable that specifies which numbering method to use in the **Value to pass to stored procedure** box.
5. If the stored procedure creates the record (for example, if you use autonumbered primary keys), select the **Stored procedure creates the record** check box. If Design Manager creates the record (based on primary key value), clear the check box.
6. From the **Edit** menu, select **Save**.

Defining the Record Properties

Similar to defining records in flat data files, you define each record in an ODBC data source data file separately. You define properties such as the start of a customer. However, since the data has not been extracted from the data source, you query each record individually. You configure properties such as which rows to include and how many.

When you define a record, unless every customer has the same number of records, you must also identify where a new customer starts.

Caution: If you do not identify the start of a customer, the engine does not reset variables to have zero elements, which can cause incorrect results at run time if you want to reset variables at the customer level.

To define the record properties:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. To create the record, do one of the following:

| To | Do this |
|---------------------------|---|
| Create the first record | If no records have been added to the data file, the ODBC Record Properties dialog box opens automatically when you drag the data file to the Edit Panel. |
| Select an existing record | Right-click the Edit Panel and select Record > Record Properties . The ODBC Record Properties dialog box opens. |
| Create a new record | Right-click the Edit Panel and select Record > Add Record . The ODBC Record Properties dialog box opens. |

3. From the **Record source** list, select **Stored procedure**.
4. In the **Rows to display** box, enter the number of rows in each column you want to appear in the Edit Panel. The default is 1.
5. If you are mapping a customer driver file and the record is the start a new customer, select the **Start new customer** check box.
6. Configure the remaining properties.
7. Click **OK**.

The **Map ODBC Columns to Variables** dialog box opens.

8. If you are creating new variables to map to the data areas, complete the properties on the dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see “[Mapping an ODBC Data Source Data File](#)” on page 342.

Defining the Input Parameters for the Stored Procedure

With the stored procedure method, you can use object identifiers to help you manage the records you write to the database. In Design Manager, you create a record for the stored procedure.

The stored procedure has two parameters: an input parameter and an output parameter. The input parameter stores the data that you want to send to the procedure. The engine uses the input parameter to insert the primary key value. The output parameter passes the data retrieved using the stored procedure to the engine. The stored procedure uses the value you select as the value to pass the value of the primary key through the input parameter. The stored procedure also uses the variable to hold the value of the output parameter after the procedure executes.

To assign a variable to an input parameter, the variable must have design sample text that retrieves at least one record from the database (using the stored procedure). Otherwise, Exstream Design and Production does not return a record and you cannot complete data mapping. During a run, the engine populates variables with values to pass to the stored procedure. However, when you map, the engine is not running so Exstream Design and Production passes the comments of the design sample to the stored procedure.

After you assign all the input parameters a value, you can view the columns and output parameters in the **Columns** tab. If you use SQL Server and the **Column** results for the stored procedure do not appear as expected in the **Selected** box, try making changes to the stored procedure. Statements such as SELECT, INSERT, UPDATE, DELETE, and PRINT can sometimes prevent SQL Server from describing the stored procedure result columns to Exstream Design and Production. To fix this, you can add the SET NOCOUNT ON command, which prevents the return of the count of rows affected by SQL statements in the procedure.

To define the input parameters for the stored procedure:

1. In Design Manager, from the Library, drag the ODBC data source data file to the Edit Panel.
2. Right-click the Edit Panel and select **Record > Record Properties**.
3. Go to the **Procedure** tab.
4. In the **Input** box, select a parameter.
5. In the **Parameter Value** box, select the variable that defines the parameter or enter a fixed input value. Enter the data consistent with the data type of the parameter (such as quotes for String, numbers for Integer, and so on).

The variable appears in the **Parameter Value** box.

6. Click **OK**.

The **Map ODBC Columns to Variables** dialog box opens.

7. If you are creating new variables to map to the data areas, complete the properties on the dialog box, otherwise click **OK**.

For more information about mapping variables to data areas, see [“Mapping an ODBC Data Source Data File” on the next page](#).

10.4 Mapping an ODBC Data Source Data File

The last step before you can begin using an ODBC data source data file is to map the data. You can automap an ODBC data source data file or you can map the data file manually the same way you do other data files.

For more information about manually mapping a data file, see “[Manually Mapping a Data File](#)” on [page 146](#).

Keep in mind that you are not required to map all of the columns returned by the query. For example, if you use a stored procedure to create a customer driver file, you are not able to change the columns or the column properties. However, you can limit the maps to just the columns you require.

The only time you can automap is immediately after you select the data to include in a data file. Automapping lets you automatically find areas of a data file, identify the type of data in the area, and map a variable. When you automap a data file, Design Manager uses the structure you created to define the data area input format and to associate a variable with each of the data areas. If you used the stored procedure method to select the data to include in the data file, you can view only the columns and parameters when mapping. Design Manager does not list input parameters since you assign variables to input parameters when you select the data.

For more information about assigning variables to input parameters, see “[Defining the Input Parameters for the Stored Procedure](#)” on [page 340](#).

To map an ODBC data source data file, complete the following tasks as needed:

- “[Automapping an ODBC Data Source Data File Using Existing Variables](#)” below
- “[Automapping an ODBC Data Source Data File Using New Variables](#)” on the next page

10.4.1 Automapping an ODBC Data Source Data File Using Existing Variables

If you want to use existing variables, the variable names and properties must match the variable names specified in the **Map ODBC Columns to Variables** dialog box.

To automap an ODBC data source data file using existing variables:

1. In the **ODBC Record Properties** dialog box, click **OK**.
The **Map ODBC Columns to Variables** dialog box opens.
2. To automatically map the variables, select the **Map variables to columns** check box.

3. Clear the **Create variables for columns** check box.
4. In the **Folder** box, select the folder in the Library in which to find existing variables.
5. Click **OK**.

The **Map ODBC Columns to Variables** dialog box closes.

10.4.2 Automapping an ODBC Data Source Data File Using New Variables

If you create variables during automapping, Design Manager uses the column properties to identify the variable type and the column name, or the output parameter name, to name the variable. Keep in mind that the variables you create during automapping must follow the Exstream Design and Production variable naming conventions.

For more information about variable naming conventions, see “[Variable Naming Conventions](#)” on [page 61](#).

To automap an ODBC data source data file using new variables:

1. In the **ODBC Record Properties** dialog box, click **OK**.
The **Map ODBC Columns to Variables** dialog box opens.
2. To automatically map the variables, select the **Map variables to columns** check box.
3. Select the **Create variables for columns** check box.
4. To add a prefix to all of the variable names that will be created during automapping, enter a string of characters in the **Prefix** box. A prefix appears as PrefixVariableName.
5. The **Column** box, select the columns or output parameters you want to map. If you selected multiple tables, the table and column information appears as TableName.ColumnName. The name of the variable appears as TableName_ColumnName.
6. To update the properties of a variable, do the following:
 - a. Click the column name.
 - b. From the **Type** list, select a data type.
 - c. If the variable is an array, select the **Array** check box. Design Manager does not detect if a variable should be an array. If the variable can have more than one value for a customer, you must select the **Array** check box.
 - d. From the **Format** list, select a data format. The options vary based on your selection from the **Type** list.
 - e. If you are mapping a customer driver file that can be updated during processing or a sort index file, select one of the following options from the **Use** list to specify the use

of the data area.

- **Input**—The data area is read from during processing.
- **Output**—The data area is written to during processing.
- **Update**—The data area is both read from and written to during processing.
- **Initialize**—The data area sets the initial value of the variable.

For more information about variable types and output formats, see “[Types of Variables](#)” on [page 60](#) and “[Formatting Variable Values in Output](#)” on [page 81](#).

Chapter 10: Setting up Data Files to Map Data in Communications Designer

To map data using the Communications Designer data source editor, you must first create a Communications Designer data file and then add it to your application.

Note: The Communications Designer data file format is available only if you have licensed the Communications Designer module.

When you run the engine, this data file is also used to identify the production data source name for the customer driver file for the Communications Designer communication.

To create a Communications Designer data file object:

1. In Design Manager, in the Library, right-click the **Data Files** heading and select **New Data File**.
2. In the **Name** box, enter a name. In the **Description** box, enter a description (optional).
3. In the **File type** list, select **Customer driver file**.
4. In the **File format** list, select **Communications Designer**.
5. Click **Finish**.

The **New Data File** dialog box closes and the print data file opens in the Property Panel for you to define.

6. On the **Basic** tab, in the **File to use in production** box, enter the symbolic name of the data source file for the engine to use during production.