# Table of Contents

**Chapter 1: Introducing Create!form Director**
- Who is this Guide For? .................................................. 10
- Create!form Director Features ........................................ 11
- Requirements .................................................................. 12
- Getting Help .................................................................. 13
- User Guide Conventions .................................................. 14
- Terminology .................................................................. 15
- Product Overview .......................................................... 16
  - Create!form Workflow .................................................. 16
  - Create!form Components ................................................. 17

**Chapter 2: Installing and Configuring** ........................................... 21
- Installation Requirements .................................................. 22
- Installing Create!form Director ......................................... 23
- Starting Create!form Director ............................................ 24

**Chapter 3: Getting Started** ............................................................ 25
- What is a Project? ............................................................ 26
- Project Directories ........................................................... 27
  - Project Directory .......................................................... 27
  - Common Project Directory ............................................. 27
- Creating a Project ............................................................ 29
  - New Project Wizard ....................................................... 29
- Input File Type ................................................................ 31
  - DataMap Options .......................................................... 31
  - Project Type ................................................................. 31
- Design Flowcharts ........................................................... 33
- Saving a Project .............................................................. 36
- Opening a Project ............................................................ 37
- Printing and Previewing Projects ....................................... 38
- Packing Projects .............................................................. 39
- Transferring Projects to the e-forms Server ......................... 40
Managing Project Files ............................................. 41

Chapter 4: Viewing and Navigating ................................. 43

Understanding the Screen Elements ............................ 44
  Design Window .......................................................... 45
  Input Window ............................................................ 45
  Input Tree ............................................................... 45
  Input File Source View ................................................ 46
  Status Bar ............................................................... 46
Navigating the Input File ............................................. 48
  Jumping Directly to a Set or Page ................................. 48
  Moving Between Sections ............................................ 49

Chapter 5: Input Design .............................................. 51

Before You Start ....................................................... 52
  Using the Create!form Sample Projects ....................... 52
  Selecting a Different Input File ................................. 52
  Viewing the Input File Source .................................... 53
Key Input Design Concepts ......................................... 54
  What is a DataMap? ................................................... 54
  What is a Set? .......................................................... 55
  What is a Section? .................................................... 55
  What is a Data Variable? .......................................... 55
  What is a Derived Variable? ...................................... 56
DataMap Types and Options ....................................... 57
  Embedded and Shared DataMaps ................................. 57
  DataMap Options When Creating a New Project ............. 58
  DataMap Options while Designing a Project .................. 58
    Edit an Embedded DataMap .................................... 59
    Copy a DataMap from Another Project ...................... 59
    Share a DataMap with Other Projects ....................... 59
    Edit a Shared DataMap .......................................... 59
    Use a Shared DataMap ............................................ 60
    Embed a DataMap ................................................. 60
Designing for CSV Input Files .................................... 62
  CSV File Structure .................................................. 62
  Input File Properties ............................................... 64
Defining Sets ................................................................. 65
Defining Sections ........................................................... 67
Working with Data Variables .............................................. 67
  Data Variables in the Input Header Section ......................... 68
  Data Variables in Detail Sections ..................................... 69
Changing the Properties of Columns ................................... 69
Changing the Input Data Format ........................................ 69

Designing for Text Input Files .......................... 70
  Input Sections .............................................................. 72
Examining how Sections are Populated ................................. 73
Design Principles ........................................................... 73
  Designing from the Top-Down ....................................... 73
  Design Flowchart .......................................................... 74
Setting the Input Page Size ............................................ 76
How the Page Size is Calculated ....................................... 78
Other Input File Settings ................................................ 78
Defining Sets ................................................................. 81
Working with Data Variables ............................................. 83
  Set Input Data Formats ................................................ 83
  Create Data Variables .................................................. 84
  Create Data Variables in a Detail Section ......................... 86
  Make Data Variables Expandable .................................... 87
Common Input Section Tasks ............................................ 88
  Display and Edit Input Section Properties ......................... 88
  Controlling When an Input Section is Generated ................. 91
  Working with Multiple Input Sections ............................... 91
Deleting Input Sections .................................................. 92
Working with the Input File Header Section ......................... 92
Working with Input Header Sections ................................... 93
  Adjust the Input Header Section Size ............................... 93
  Make an Input Header Section Expandable .......................... 94
  Add Input Header Sections ............................................ 96
  Rules for Creating Multiple Input Header Sections ................ 98
Working with Detail Sections ............................................. 98
  About Defining Detail Sections ..................................... 98
  Add Detail Sections .................................................... 100
  Make a Detail Section Expandable .................................. 101
  Rules for Creating Multiple Detail Sections ....................... 102
Working with Input Footer Sections ................................. 102
  Adjust the Input Footer Section Size ............................... 102
  Add Input Footer Sections ............................................ 103
  Rules for Creating Multiple Input Footer Sections ............... 104

Designing for XML Input Files .......................... 105
XML File Structure ....................................................... 105
Input File Properties ..................................................... 107
Defining Sets ................................................................. 108
Defining Sections ......................................................... 110
Working with Data Variables ......................................... 110
  Changing the Properties of Data Variables ........................................ 111
  Changing the Input Data Format ........................................ 112
Designing for JDE PDF Input Files .............................. 113
Input File Properties .................................................. 114
  Defining Sets .......................................................... 115
  Input Sections .......................................................... 117
  Data Variables .......................................................... 117
Creating Derived Variables ........................................ 118

Chapter 6: Nodes ............................................................. 119

  Key Design Concepts .................................................. 120
    About Nodes ........................................................ 122
    About Parent, Sibling and Child Relationships .......................... 124
    Sequence and Timing .............................................. 126
    Using Variables and Expressions to Control Actions .................. 126
  Adding Nodes .......................................................... 128
  Selecting and Editing Nodes .......................................... 130
    Select Nodes ...................................................... 130
    Insert Nodes ...................................................... 130
    Edit Nodes ........................................................ 132
    Move Nodes ....................................................... 133
    Copy and Paste Nodes ............................................ 133
    Delete Nodes ...................................................... 135
  Node Properties ........................................................ 136
    Send Node ......................................................... 136
    Preview Node .................................................... 141
    Condition Node ................................................... 142
    Split Node ........................................................ 142
    Sort Node .......................................................... 143
    Sub Project Node .................................................. 144
    Merge Node ........................................................ 145
    Transform Node .................................................... 146
    Update Node ....................................................... 146
    Get File Node ...................................................... 147
    Delete File Node ................................................... 148
    Run Program Node ................................................ 149
  Window Display Preferences ........................................ 150
Chapter 9: Expressions and Conditions 187

Expressions 188
Building Expressions 188
Using the Expression Builder 188
Navigating and Editing Expressions 189
Conditions 190
Building Conditions 190
Using Expressions as Conditions 191
Building Blocks 192
Boolean Operators 192
Operators 193
Functions 195
Variables 200
System Conditions 201
Syntax and Order of Precedence Rules 202
Mixed Data Types 202
Addition Operator 203
Subtraction Operator 203
Comparing Text Strings 203
Order of Precedence 204
Nested Functions 204
Parsing 204
Names of Variables 205

Chapter 10: Customizing 207

About User Preferences 208
General Preferences 210
Section Borders 210
Input Source Preferences 211
Design Window Preferences 212
Design Tree 212
Input Window Preferences .............................................. 214
CSV ........................................................................... 214
JDE PDF ...................................................................... 214
JDE PDF Sections ........................................................ 215
Text ............................................................................. 216
Text Sections ............................................................... 217
XML ........................................................................... 218

Appendix A: Glossary ..................................................... 219

Glossary ........................................................................ 220
INTRODUCING CREATE!FORM DIRECTOR

Create!form Director is a design component of the Create!form™ Distributed Output Management (DOM) system that allows you to prepare the output from your enterprise server for merging, emailing, faxing and archiving.

The following topics are covered in this chapter:

- Who is this Guide For?
- Create!form Director Features
- Requirements
- Getting Help
- User Guide Conventions
- Terminology
- Product Overview
INTRODUCING CREATE!FORM DIRECTOR

Who is this Guide For?

This user guide is for people who design projects with Create!form Director.
Create!form Director Features

With Create!form Director you can customize your documents without making modifications in your enterprise server. The following examples introduce some of the features that Create!form Director offers.

Printing, Faxing and Emailing
Direct the printer output from your enterprise server to multiple, customized merge forms that are selected automatically based on who generated the output and the contents of the output. The merged documents can then be duplicated and sent to multiple destinations for print, email, fax and archive.

Transactions
XML transaction data can be captured and the data presented in graphical forms and published to a URL, and emailed to individual email addresses, and transformed into CSV format for use in a spreadsheet or database.

Databases
Data from any database can be extracted and merged with other documents as the output is generated for print, email, fax and archive. Databases can also be updated using data captured from transactions and the printed or file output of any third party application.

Archives
When information is printed, faxed, emailed, added to databases, stored on disk, or published to a web site, the same data can be automatically archived in the same or a different form and format.
Requirements

The following are required before you can begin designing your project:

Prerequisites

- Basic familiarity with Windows concepts
- Basic familiarity with word-processing or publishing software
- Familiarity with report printing on your enterprise server

Design Workstation

For information on system and hardware requirements, refer to Chapter Two, Installation Requirements.

Enterprise Server

Create!form Director can work with the following types of spool files:

- Plain ASCII text files (with or without a job header).
- CSV data files
- XML transactional data files
- PeopleSoft EnterpriseOne reports (JDE PDF): B73.3.2 (cum 2) SP14.2 or B73.3.3 (Xe) and above.

Depending on the capabilities of the program used, it may be possible to configure the server application to attach the job header parameters to the spool file, or you may need to install the Create!send utility, which is available for sending spool files from OS/400, UNIX, Windows and VMS platforms.

e-forms Server

Before you can start printing, you must install and configure Create!form Server on the network e-forms server. For more information see the Create!form Server user guide. Depending on your preferred output format, you may also need to install other Create!form components. You will find an overview of how to install and configure your e-forms production system in the implementation road map in the Create!form Server user guide.
Getting Help

Help is available in both Portable Document Format (PDF) and as online help.

To view and print the PDF user guide
- Open the file Create!form Director 6.3.pdf with Adobe Reader from the \Manuals directory on the installation CD
- From the Windows Start menu, select:
  Programs > Create!form > Manuals > Create!form Director

To view the online help
- Start Create!form Director and select Help>Help Topics from the Main menu.
User Guide Conventions

Section headings are listed at the front of each chapter to assist you in finding the relevant information. In the electronic version of this document, clicking on these and other cross references will take you to the appropriate section. The following typographical conventions are used:

Table 1.1:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Convention Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bold</td>
<td>Screen items and buttons</td>
</tr>
<tr>
<td>cross reference</td>
<td>Short cut or reference to the document, chapter or section</td>
</tr>
<tr>
<td>NOTE</td>
<td>Additional information relating to the topic.</td>
</tr>
<tr>
<td>Item &gt; Sub-item</td>
<td>Menu selection; i.e., select the menu Item and then select the Sub-Item from the drop down list, e.g. File &gt; New.</td>
</tr>
<tr>
<td>Numbering 1</td>
<td>Step-by-step instructions. To perform an action, follow these instructions in the sequential order given.</td>
</tr>
</tbody>
</table>
Terminology

Throughout this user guide, you will see references to many different terms, some of which may be new to you. To familiarize yourself with the terminology used in this manual, please read the definitions provided in Appendix A, Glossary.
Product Overview

What would you like to know about?

• Create!form Workflow

• Create!form Components

Create!form Director is part of the Create!form Distributed Output Management (DOM) system: a suite of products that allow users to prepare, format, produce, distribute and administer the transactional and document output of your enterprise server.

Create!form Workflow

Create!form operates in conjunction with the print spooler, native to the installation platform. The output from your enterprise server is spooled to a Create!form Server print queue for processing. How the job will be processed is determined by header information, which can be added to the spool file by the application, or by sending the job initially to specially configured print queues, which tag and route the job to the appropriate Create!form Server print queue. This process of tagging and routing jobs is performed by Create!send.

Create!form Server monitors the jobs submitted to print queues, intercepting any jobs that are tagged for processing. There are three types of projects that Create!form Server can apply to the spool file, which are built with Create!form companion products:

• The spool file can be pre-processed and split into components by applying projects created with Create!form Director.

• The spool file can be restructured and converted into a different format like a CSV or XML by applying projects created with Create!form Transform.

• The spool file can be merged with customized forms, known as form projects, by applying projects created with Create!form Designer.

As jobs are submitted to different print queues, Create!form Server determines which project it should be processed with and builds an output file that can then be printed, faxed, e-mailed or archived using other Create!form products – as described in Figure 1-1. The output file format is typically PostScript or PDF, depending on its intended destination, but other formats are supported.
Create!form Components

The following Create!form products may be installed:

**Create!form Designer**

Create!form Designer is the product you use to design form projects, which Create!form Server merges with the output of your enterprise server. The
form project allows you to reformat and enhance the output of your enterprise server, incorporating logos, images and drawing objects, as well as formatted text.

Create!form Transform
Create!form Transform is the product you use to design projects with which Create!form Server transforms the output of your enterprise server. The project allows you to restructure and reformat the output of your enterprise server, which may include text, CSV, XML and other file formats.

Create!form Director
Create!form Director is the product you use to design projects with which Create!form Server splits and processes the output of your enterprise server, ready for merging, transforming, emailing, faxing and archiving.

The types of action that Create!form Director performs include:
• Splitting the spool file into multiple spool files
• Sending particular parts of the spool file to particular directories or queues
• Extracting data for use in reports
• Creating backup copies of the spool file
• Adding job ticket templates (JTTs) that determine various parameters used in processing by other Create!form products

Create!form Server
Create!form Server is the central processor, which handles the output from your enterprise server, and applies the appropriate project designed with either Create!form Designer, Create!form Transform or Create!form Director. The output from Create!form Server can be printed, faxed, e-mailed or archived using other Create!form products.

Create!print
Create!print is called by Create!form Server to print the merged documents, and is installed automatically with Create!form Server on the e-forms server. Create!print is controlled and activated by Create!form Server without user interaction. A Create!print license is required for each print queue/device. To obtain additional Create!print licenses, contact your Create!form distributor.

Create!send
Create!send is a utility that inserts job information in the spool file as it is printed from your enterprise application. You can use Create!send to insert job header parameters that allow Create!form Server to process the job.
Create!send is available for sending spool files from OS/400, UNIX, Windows and VMS platforms.

**Create!email**
Create!email lets you automatically e-mail the merged documents. For more information on Create!email, contact your Create!form distributor.

**Create!fax**
Create!fax lets you automatically fax the merged documents. For more information on Create!fax, contact your Create!form distributor.

**Create!archive**
Create!archive lets you automatically archive the merged documents on an intranet or the internet, where they can be accessed under secure, password control. For more information on Create!archive, contact your Create!form distributor.

**Create!form Bind**
Create!form Bind is an optional module that takes a collection of files, typically a collection of Invoices and Purchase Orders for a single client, and produces a single PDF. The bound document can then be sent to Create!form Server for processing.

**Create!micr**
Create!micr lets you print documents containing MICR (Magnetic Ink Character Recognition) text, which is used in the banking industry for check/cheque printing. The evaluation version of Create!micr can be installed from the installation CD.

**Create!pdf server**
Create!pdf server lets you create documents in PDF output format, which is typically required for sending documents to Create!email, Create!archive and PDF enabled print devices. Create!pdf server is installed with the Windows version of Create!form Server and requires a separate license. To obtain a license for Create!pdf server, contact your Create!form distributor.
20 INTRODUCING CREATE!FORM DIRECTOR

- Product Overview

-
INSTALLING AND CONFIGURING

Create!form Director is installed on a Windows workstation from the Create!form installation CD.

The following topics are covered in this chapter:

• Installation Requirements
• Installing Create!form Director
• Starting Create!form Director
Installation Requirements

Create!form Director requires the following:

- Microsoft Internet Explorer 5.0 or greater
- Adobe Reader 4 or greater (for viewing the User Guide)
- approximately 30 MB of free hard disk space
- a PostScript level 2 or greater printer is recommended
Installing Create!form Director

You will find an overview of how to install and configure your e-forms production system in the implementation road map in the Create!form Server user guide.

For more information on upgrading an earlier version, see the release notes located in the \Readme directory of the installation CD.

To install Create!form Director

1. Insert the installation CD. The installation screen should be displayed automatically. If it isn’t, run the CForm.exe program located in the root directory of the installation CD.

2. Select the Install Products button, and then select Create!form Director from the menu.

3. If no other Create!form product is installed, you will be prompted to select the installation directory. If not, the installation directory is already defined.

4. Specify the default location for saving your project files by specifying the project directory. By default, the WorkDir directory in the Create!form Director installation directory will be used. If this is the first Create!form product you have installed, it is recommended that you accept the default setting. For more information about Project Directories, refer to Chapter Three, Project Directories.

5. The Setup program allows you to choose between a Typical or a Custom installation.
   • Select Typical to automatically install the program, all help files, sample files, and the English-UK and English-US spelling dictionaries.
   • Select Custom to install specific options such as particular spelling dictionaries.

6. After specifying the location for the program icon on the Start menu, verify that all settings are correct and click Next to complete the installation.

7. Return to the main menu and click Exit when you have finished installing Create!form products.
Starting Create!form Director

From the Windows Start menu
Click the Start menu and select:
  Programs > Create!form > Create!form Director 6
For information on screen elements and toolbars refer to Chapter Four, Viewing and Navigating.
For information on getting started with Create!form Director refer to Chapter Three, Getting Started.
Getting Started

Your design is recorded in a project, which contains information about how the input data will be mapped and processed. Each project consists of several files, located in a number of folders, which are required whenever the project is opened, saved, moved or transferred.

The following topics are covered in this chapter:

- What is a Project?
- Project Directories
- Creating a Project
- Design Flowcharts
- Saving a Project
- Opening a Project
- Printing and Previewing Projects
- Packing Projects
- Transferring Projects to the e-forms Server
- Managing Project Files
What is a Project?

A project is a record of the way you want to process input data files. Because your enterprise server may output a variety of different documents, you will need to create a project for each of these variations.

A project consists of an input design, which describes the structure of the input file, and an output design, which describes how the input file will be processed. The input design will be displayed in the Input Window, and the output design will be displayed in the Design Window.
Project Directories

The project files are stored in the following folders:

- Project Directory
- Common Project Directory

The project also uses resources from a number of system folders.

Project Directory

Each project has a project directory. This is the location where you save the project and where its components are stored. When Create!form Director is installed, a default project directory is created: `<install dir>\WorkDir`. This is where you will be prompted to save all new projects. During the installation or at a later time, you can change this default project directory.

To change the default project directory

1. From the Main menu, select Tools > Preferences. The Preferences dialog will be displayed.
2. Set the Default Project directory.
3. Click OK.

It is recommended that you use separate directories for design projects and production projects. For example, use `WorkDir` for your design work and create a new directory `ProdDir` for completed, live projects.

Common Project Directory

Project components that are shared between projects are stored in the common project directory. When Create!form Director is installed a default common project directory is created: `<Install dir>\CommonProject`. At a later time, you can specify another location for the common project directory.
To change the common project directory

1. From the Main menu, select Tools > Preferences. The Preferences dialog will be displayed.

2. Set the Common project directory.

3. Click OK.
Creating a Project

Before creating a project, you must choose a suitable input file. The input file should be representative of the files likely to be encountered in a production environment and should contain all possible variations of content and format. The elements and structures of the input file, which you identify during the input design process, are recorded in a DataMap. For more information on creating a DataMap refer to Chapter Five, Input Design.

The New Project Wizard will guide you through the process of creating a new project. You will be given a number of options by the wizard, which may include:

- Input File Type
- DataMap Options
- Project Type

To complete your design, follow the steps described in the Design Flowcharts relevant to the input file type you have selected.

New Project Wizard

The New Project wizard will guide you through the process of creating a new project.

To start the New Project wizard

1. From the Main menu, select File > New, or click the New Project button. The New Project Wizard dialog will be displayed.
2. From the Type drop-down, select the type of source file from the available file types.
3. From the File/Data source drop-down, select the input source file you wish to open, from the current project directory. If the file you want is not listed, click to display the Select Input File dialog. Locate the input file from this dialog and click Open to select the file.
4. Click Next. The sequence and content of dialogs displayed by the wizard will vary depending on the type of input file selected in the first dialog.
5 Continue to follow the wizard instructions and click Finish when you have completed all options.

For descriptions of these options, refer to:
- DataMap Options
- Project Type
Input File Type

The New Project wizard will give you the option of selecting from the available input file types. The input file types supported by Create!form include text, CSV, XML and others. For more information refer to Chapter Five, Input Design.

DataMap Options

The New Project wizard gives you the option of creating a new DataMap, copying the DataMap from another project, or using a shared DataMap.

Create a new DataMap

- Choose the Create a new DataMap option if you have not previously created a DataMap for an input file of this type and structure.

If a suitable DataMap does exist, you can choose to either copy the DataMap, or, if it has already been shared, you can share the DataMap with other projects.

Copy a DataMap from another project

Choose the Copy a DataMap from another project option if you need to modify the DataMap in any way.

Select a shared DataMap

Choose the Select a shared DataMap option if the DataMap can be used without modification.

For a full description of these options refer to Chapter Five, DataMap Types and Options.

Project Type

When you are working with a text input file, you can choose the type of project you want to create. The New Project wizard gives you the option to create either a standard or a simple project:
**GETTING STARTED**

- *Input File Type*

---

**Standard Project**

Create a standard project when the input page structure is variable. In a standard project you can utilize Create!form Director’s many features to map any data from the input file to the project, and then manipulate the input data in the project.

**Simple Project**

Create a simple project if the input page structure is uniform. In a simple project, the entire input page is mapped as one section, which you can selectively map as data variables.
Design Flowcharts

The following flowcharts list the steps required to create projects for the main input file types.

CSV

Figure 3.1: An overview of the design processes for CSV
Set the Input Page Size
(Refer to Setting the Input Page Size for details.)

Define Sets
(Refer to Defining Sets for details.)

Standard Project:
Create a DataMap
(Refer to Input Design for details.)

Simple Project:
(Refer to Create Data Variables for details.)

Create a Process Design
(Refer to Chapter 6, Nodes, for details)

Figure 3.2: An overview of the design processes for text input files
**XML**

- Define Sets
  (Refer to Defining Sets for more details.)

- Define Input Sections
  (Refer to Defining Sections for more details.)

- Create a Process Design
  (Refer to Chapter 6, Nodes, for more details.)

*Figure 3.3: An overview of the design processes for XML input files*

**JDE PDF**

- Define Sets
  (Refer to Defining Sets for more details.)

- Create a Process Design
  (Refer to Chapter 6, Nodes, for more details.)

*Figure 3.4: An overview of the design processes for JDE PDF input files*
Saving a Project

You should regularly save your work. You can store projects in any directory, but Create!form Director looks in only two directories for the files used in each project:

- the `<Install dir>`\CommonProject directory, which typically contains items common to various projects; or
- a project directory dedicated to particular projects. You can create as many project directories as you want. On install, one project directory exists, called `<Install dir>`\Workdir.

Note

When moving a project to another location, save it as a packed project file type. Refer to Packing Projects for more information.

To save a project

From the Main menu, select File > Save, or click on the Main toolbar. If you are saving a project for the first time, the Save As dialog will be displayed.

Unless you have selected the Save to common project directory checkbox, Create!form Director will save the project and the DataMap to the current project directory. For more information about the files created by Create!form Director refer to Managing Project Files.
Opening a Project

To view, edit, print or transfer a project, you must open the project with Create!form Director.

To open a project

1. From the Main menu, select File > Open or click from the Main toolbar. The Open dialog will be displayed and show files from the current project directory. To display files in the \CommonProject directory, select the Look in common project directory checkbox.

2. Select the project and click Open.
Printing and Previewing Projects

Project designs can be previewed on-screen and printed locally to a PostScript printer.

To print a project
1. From the Main menu, select File > Print. The Print dialog will be displayed.
2. Select the required printing options.
3. Click OK.

To preview a project
From the Main toolbar, click on the Main toolbar or select File > Print Preview from the Main menu.
Packing Projects

Because projects are made up of several different files (the project file, input data file, DataMap, etc.), Create!form Director allows you to save all associated files into a single packed project. The entire project can then be easily moved and opened on any computer.

**To save a project as a packed project**

1. Open the project.
2. From the Main menu, select File > Save As. The Save As dialog will be displayed.
3. In the Save as type field, select Packed director project files (*.dpp6).
4. Click Save. The packed project is saved with the specified name, with an .dpp6 extension. To send the packed project, simply email it as an attachment.

To open a packed project, from the Open dialog, select Packed director project files (*.dpp6) from the Files of type field.
Transferring Projects to the e-forms Server

Before the project can be used for production printing, you must transfer the project and its associated resources to the e-forms server. You will find a complete discussion on transferring projects in the Create!form Server user guide.

To transfer a project to the e-forms server

1. From the Main menu, select Tools > Transfer. The Transfer to Server dialog will be displayed.

2. Select the project to be transferred and its destination.

You will find a complete discussion on transferring projects and the options displayed on the Transfer to Server dialog in the Create!form Server user guide.
Managing Project Files

Create!form Director creates and uses a number of different file types that can be identified by their icons and filename extensions.

<table>
<thead>
<tr>
<th>File Type</th>
<th>Icon</th>
<th>Extension</th>
</tr>
</thead>
<tbody>
<tr>
<td>Project</td>
<td><img src="image" alt="CDP6 Icon" /></td>
<td>.cdp6</td>
</tr>
<tr>
<td>Project backup</td>
<td><img src="image" alt="CDP6Bak Icon" /></td>
<td>.cdp6bak</td>
</tr>
</tbody>
</table>
| DataMap          | ![CSV Icon](image) | .dmp6(text)
|                  |      | .cmp6(CSV)
|                  |      | .xmp6(XML)
|                  |      | .jmp6(JDE PDF) |
| DataMap backup   | ![CDP6Bak Icon](image) | .dmp6bak etc |
| Packed project   | ![DPP6 Icon](image) | .dpp6     |
| Input file       | various | external source |

The project files may be located in either the project directory or the common project directory.

Care should be taken when deleting, moving or renaming project files as this may corrupt the project. Whenever you move, copy or send a project to another location, you should pack the project first.
GETTING STARTED
- Managing Project Files
-
VIEWING AND NAVIGATING

Create!form Director allows you to interact with and navigate through your projects with a familiar, graphical and easy to use interface, which can be customized to suit your needs.

The following topics are covered in this chapter:

- Understanding the Screen Elements
- Navigating the Input File
Understanding the Screen Elements

The Create!form Director interface can be customized to suit your needs. The following graphic shows the basic screen elements.

Note that the appearance and position of each element may vary depending on screen size and the current view and customization settings.

*Figure 4.1: The Create!form Director screen*
The following topics describe the function of the different windows and views in detail.

**Design Window**

The Design Window displays the project design as a flowchart or process tree consisting of nodes and connectors. The nodes represent actions and destinations by which the input file can be split, sorted, merged, printed, copied and otherwise processed. These nodes are created by selecting tools from the Design toolbar. When you float the cursor over nodes in the design, information about the type of process will be displayed.

You cannot hide the Design Window, but you can customize its look and feel. For more information refer to Chapter Ten, Design Window Preferences and Design Tree.

**Input Window**

The Input Window displays the input design. When you float the cursor over data in the Input Window, the name of the data variable will be displayed (text and JDE PDF input files only).

To show or hide the Input Window, click the Input Window on the Main toolbar or select View > Input Window from the Main menu. Several task shortcuts are available by right-clicking on items in the Input Window.

You can customize the look and feel of the Input Window.

**Input Tree**

The Input Tree displays the logical structure of the input file. It provides details about the input file as a whole and individual sections within it. Several task shortcuts are available by double-clicking and right-clicking on items in the Input Tree.

To show or hide the Input Tree, click the Input Tree on the Main toolbar or select View > Input Tree from the Main menu.

You can customize the look and feel of the Input Window. For more information refer to Chapter Ten, About User Preferences.
Input File Source View

The Input File Source View displays the raw input file.
To show or hide the Input File Source View, select View > Input File Source from the Main menu.

Status Bar

The Status bar provides a variety of information about the project. The information displayed will depend on the current active screen element.

When the Input Window is active

Tabs are displayed for each of the section types on the current page. The selected section is highlighted.

The page number in the input file and the total number of pages.
The page coordinates of the cursor.

The size of currently selected section.
The page coordinates of the cursor.

The number of the set and the total number of sets in the project.
The page number of the set and the total number of pages in the set.
The type of section currently selected.
The name of the section the cursor is currently above.

Figure 4.2: Information displayed in the Status bar when the Input Window is active
When the Design Window is active

| Ready | Condition | 100% | Run Program |

The type of node currently selected.

The type of node the cursor is currently above.

*Figure 4.3: Information displayed in the Status bar when the Design Window is active*
Navigating the Input File

The section tabs on the Status bar only appear when the Input Window is active, and only sections on the current page are displayed. To locate occurrences of some sections, you may need to browse the pages of the input file.

To navigate to other pages or sections of the input file
Click the appropriate button from the Navigation toolbar. The Input Window must be active:

![Navigation toolbar buttons]

*Figure 4.4: The buttons on the Navigation toolbar*

These commands are also available through the main menu.

Jumping Directly to a Set or Page

To go to a specified page number
1. From the Main menu, select Page > Go To. The Go To Page dialog will be displayed.
2. Enter the page number to go to, or select the page number from the drop-down.
3. Click **OK**. The page selected will be displayed.
Moving Between Sections

In order to edit sections, they must be selected. To select sections, the Input Window must be current. You can select sections in the following ways:

- Click on the section
- Press the arrow keys to move up or down one section
- Press the Tab key to move to the next occurrence of that section and Shift+Tab to move to the previous occurrence of that section
- Click on the section tab on the Status bar; all sections that are displayed on the current page will have tabs
- Select the section in the Input Tree
- Use the buttons on the Navigation toolbar
VIEWING AND NAVIGATING
- Navigating the Input File
Chapter 5: Input Design

The input design process involves identifying and labelling the different parts or elements of the input file. The way in which you identify, label and organize the data in the input file is recorded in the DataMap.

The following topics are covered in this chapter:

- Before You Start
- Key Input Design Concepts
- DataMap Types and Options
- Designing for CSV Input Files
- Designing for Text Input Files
- Designing for XML Input Files
- Designing for JDE PDF Input Files
- Creating Derived Variables
Before You Start

The following topics contain information about some basic procedures, which you may find useful:

- Using the Create!form Sample Projects
- Selecting a Different Input File
- Viewing the Input File Source

Before proceeding to the details of designing for a particular file type, you must also be familiar with:

- Key Input Design Concepts
- DataMap Types and Options

Using the Create!form Sample Projects

Sample input files have been provided with the installed software so that you can practice performing the tasks described in this user guide. The samples can be found in the <install dir>\CF6Samples\Tutorials folder. You can view the text, CSV and XML sample files with any text editor. You should complete each task in sequence as later tasks may assume that earlier tasks have already been completed.

Selecting a Different Input File

If the input file you chose for the input design is unsuitable, or if you want to test your input design on another sample file, you can select a new input file at any time.

To select a different input file

1. From the Main menu, select Project > Select Input File.
2. The Select Input File dialog will be displayed.
3. Navigate to the folder where the file is located and select the new input file.
4. Click Open. The project will refresh with the new data.
Viewing the Input File Source

To view the input file source, select the View > Input File Source command from the main menu.
Key Input Design Concepts

Before you begin your design, you must be familiar with the following key concepts:

- What is a DataMap?
- What is a Set?
- What is a Section?
- What is a Data Variable?
- What is a Derived Variable?

What is a DataMap?

The input design process involves identifying and labelling the different parts or elements of the input file such as header data, line items and footer text. The input file may be an unstructured text file, for which you must create rules that identify each part of the file, or the file may be structured, like the CSV and XML formats, which are largely self-defining, but may still require some editing or relabelling. The way in which you identify, label and organize the data in the input file is recorded in the DataMap. You must define a DataMap for every project; however, where the input file structure is the same, you can share the same DataMap between several projects.

The information recorded in the DataMap will include:

- The properties of the input file
- How sets are defined
- How sections are defined
- How data variables are defined
- How derived variables are defined

For more information about creating, editing and sharing DataMaps refer to DataMap Types and Options.
What is a Set?

Where the input file contains sequential related pages or blocks of data, such as account statements for multiple customers, you can group these pages or data together as sets.

During production printing, each set can be treated as a self-contained document. Defining sets also allows you to make use of various pre-defined system conditions, which can be used to apply different formats, or insert extra pages at the start and end of each set.

What is a Section?

A section describes a repeated group of data in the input file. For example, in a text file, the rows that contain header text at the top of each page will form a section, while in a CSV file, each row will form a section.

By describing all the different types of data in the input file in this way, the information contained in each section can be more easily reorganized and reformatted in the project.

What is a Data Variable?

Input sections can be composed of several different types of information, for example, the input header sections might contain names, addresses, dates and various reference numbers. A data variable specifies a particular field of data in each section.

Input data cannot be displayed or used in the project unless the data has been identified as a data variable in the input file. The process of defining data variables will vary according to the type of input file you are processing. For more information, refer to the relevant input format type in the sections that follow.
What is a Derived Variable?

You can combine and manipulate data variables, in user defined expressions that are called derived variables. Examples of how derived variables can be used include:

- to retrieve a substring of characters from a data variable
- to concatenate two or more data variables into one string of characters
- to convert the format of dates
- to find abbreviations and substitute with the expanded text
- to derive values from an arithmetic formula using one or more data variables

Derived variables will appear as data variables in the project. The properties of derived variables and how they are defined are described in Creating Derived Variables.
DataMap Types and Options

You can use either an embedded DataMap, or share a DataMap with other projects, which can be selected when you create the project, or later while designing the project.

What would you like to know about?
- Embedded and Shared DataMaps
- DataMap Options When Creating a New Project
- DataMap Options while Designing a Project

Embedded and Shared DataMaps

A DataMap can be either:
- **Embedded**
  An embedded DataMap is associated with a single project, and can be edited and saved without reference to other projects. When you create a new DataMap, or copy a DataMap from another project, the DataMap status is set to embedded.
- **Shared**
  A shared DataMap can be used by multiple projects. Any changes made to a shared DataMap will affect every project that uses it.

To confirm the status and name of a DataMap

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. The status of the DataMap is displayed, and
   - if the status is shared, the name of the shared DataMap is also shown.
   - if the status is embedded, the name of the DataMap can be determined by adding the appropriate extension to the stem of the project name e.g. if the input file type is CSV, the project MyProject.cdp6 creates an embedded CSV type DataMap with the name MyProject.cmp6. For more information about file names refer to Managing Project Files.
DataMap Options When Creating a New Project

The New Project wizard gives you the option to:

- **Create a new DataMap**
  Choose this option if you have not previously created a DataMap for the type of input file you are working with.

- **Copy a DataMap from another project**
  Choose this option if you have previously created a DataMap in another project for the same input file, which you want to modify for the new project.

- **Select a shared DataMap**
  Choose this option if you are able to use a DataMap from another project without modification. You must Share a DataMap with Other Projects before it can be selected.

DataMap Options while Designing a Project

In an existing project, you can edit the DataMap or change the status of the DataMap in several ways:

**What do you want to do?**

- Embed a DataMap
- Copy a DataMap from Another Project
- Share a DataMap with Other Projects
- Edit a Shared DataMap

---

**Note**

You must exercise care when changing the DataMap in an existing project, to ensure that the input sections, data variables and derived variables defined in the new DataMap are compatible with your existing design.
**Edit an Embedded DataMap**

You can edit an embedded DataMap at any time. The DataMap will change whenever the input design is changed.

**Copy a DataMap from Another Project**

To copy a DataMap from another project

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. Select the Copy from another DataMap option.
3. Click Select and locate the DataMap file you want to copy.
4. Click OK.

The existing DataMap will be overwritten and the status remains embedded.

**Share a DataMap with Other Projects**

To share a DataMap with other projects

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. Select the Share this DataMap as option.
3. The name of the shared DataMap is displayed, and can be edited.
4. Click OK.

The DataMap will be moved to the \CommonProject directory and renamed. The DataMap status will be changed to shared.

**Edit a Shared DataMap**

You can edit a shared DataMap from any project that uses the DataMap. When you edit a shared DataMap, the changes will be applied to all projects that use that DataMap. A warning will be displayed before the edited DataMap is applied to other open projects.
Use a Shared DataMap

Before you can use a shared DataMap, you must first share the DataMap from the original project. For more information on how to share a DataMap refer to Share a DataMap with Other Projects.

To use a shared DataMap

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. Select the Select a shared DataMap option.
3. Click Select and locate the DataMap file you want to use. Shared DataMaps are stored in the project and common project directories.
4. Click OK.

The DataMap status will be changed to shared.

To select a different shared DataMap

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. Select the Select a different shared DataMap option.
3. Click Select and locate the DataMap file you want to use. Shared DataMaps are stored in the project and common project directories.
4. Click OK.

Embed a DataMap

In a project that uses a shared DataMap, you can embed either the shared DataMap, or a DataMap from another project.

To embed a DataMap

1. From the Main menu, select the Project > Change DataMap command. The Change DataMap dialog will be displayed.
2. Select one of the following options:
   - Embed DataMap into project
   - Embed copy of another DataMap into project
3 If selecting another DataMap, click Select and locate the DataMap file you want to use.

4 Click **OK**.

The DataMap is copied into the Project Directory and named according to the project stem and DataMap type (For more information refer to *Managing Project Files.*) The DataMap status will be changed to embedded.
Designing for CSV Input Files

The input design process for CSV files involves identifying the row and column structure of the CSV input file; this information is recorded in the DataMap. A DataMap for an CSV input file is denoted by the .cmp6 file name extension.

What would you like to know about?

• CSV File Structure
• Input File Properties
• Defining Sets
• Defining Sections
• Working with Data Variables

To create the CSV sample project

1 To start the wizard, click the New Project button on the Main toolbar.
2 From the Type drop-down, select CSV from the available file types.
3 Click to display the Select Input File dialog and select the file Chapter5Sample.csv from the <install dir>\CF6Samples\Tutorials folder.
4 Click Next.
5 Select the Create a new DataMap radio button and click Next. A new DataMap will be created.
6 Click Finish. The new sample project will be displayed in the Input Window.

CSV File Structure

A CSV file contains repeated rows of data, with each field in a row separated by a particular character called the delimiter. Typically fields are separated by a comma, but other characters can be used, including spaces and tabs. The file may also contain a header section, and a column heading row. The important elements of a CSV file are demonstrated in sample comma delimited file below:
Note that text values that contain the delimiter character are enclosed by quotation marks, which is referred to as the text qualifier.

**Paginated CSV Files**

If the CSV input file is paginated and contains page header and footer sections, you must treat the input as a text file, in which you will be able to define the page length and separately identify the detail sections from the header and footer sections. In a text input file, you can create a data variable that contains the entire data row, and then use the retrieve function to extract the delimited field values as user variables. For more information refer to Designing for Text Input Files.
Input File Properties

The input file properties determine how the structure of the CSV input file will be interpreted.

To change the input file properties

1. Select the Input Window by clicking anywhere inside the input window frame.

2. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.

3. In the Delimiter section, select a delimiter character and a text qualifier.

4. In the Header section, select the number of header rows and indicate whether the file contains a column heading row.

5. In the Data format section, select the format used for date and decimal values in the input file.
6 Click **OK**.

The changes will be applied to the input file and the input window will be refreshed.

**Defining Sets**

You can use the following means to determine when a new set starts:

- The entire input file can be treated as one set (default setting)
- When the value in one or more columns changes
- Or, one set per row

In addition, you can now add conditional settings for each new set, using three options to define Child Tag values.
To define sets

1. From the Main menu, select Set > Define.

2. On the Define Set screen, select a desired tag:
   - Treat entire spool file as one set.
   - When values change in marked columns:
     Select one or more columns.
   - One set per row
In the Conditional set logic section, you can also configure additional settings contingent on Child Tags values.

- In the Condition set logic section, click Conditionally define sets to activate the Child Tags field box.
- Highlight the Child Tag you want to work with and select one of the desired radio button options. (For example, if you are working with an Invoice set, you might want to select invoice_number and click the “When value changes” radio button).

4  Click OK.

**Defining Sections**

Sections are created automatically from the structure defined by the Input File Properties dialog (Refer to Input File Properties for more information.):

- **InputHeader1**
  The optional section at the top of the file that contains header text

- **Detail**
  The repeating rows in the body of the file that contain the delimited data

The sections created are displayed in the Input Window.

**Working with Data Variables**

Data variables are created automatically from the structure defined by the Input File Properties dialog. The data variables created are shown in the Input Tree view:
What would you like to know about?

- Data Variables in the Input Header Section
- Data Variables in Detail Sections
- Changing the Properties of Columns
- Changing the Input Data Format

Data Variables in the Input Header Section

In the input header section, a data variable will be created for each header line.
Data Variables in Detail Sections

In detail sections, a data variable will be created for each column. The column heading will be used as the data variable name.

Changing the Properties of Columns

You can change the name, type and other properties of columns that define data variables.

To change the properties of a column

1. In the Input Window, click the column heading you wish to edit. The Column Properties dialog will be displayed.
2. To change the column heading and the name of the data variable, type a new name in the Name field.
3. In the Type field, select text, numeric or date as the data type.
4. To repeat the last non-blank value in blank fields, select the Repeat previous value if blank checkbox.
5. Click OK.

Changing the Input Data Format

Create!form Director interprets dates and numeric data in the input file using the data formats defined by the input file properties. For example, you can change the decimal separator to a comma, or the date order from year-month-day to month-day-year.

To change the input data format

1. From the Main menu, select Input > File Properties.
2. The Input File Properties dialog will be displayed.
3. From the Date order drop-down, select the required date format.
4. From the Decimal symbol drop-down, select the required numeric format.
5. Click OK.
Designing for Text Input Files

The input design process for text files involves identifying and labelling the different parts or sections of the input file such as header text, line items and footer text. You must create rules that identify each of these parts of the input. These rules are recorded in the DataMap. A DataMap for a text input file is denoted by the .dmp6 file name extension.

The information recorded in the DataMap includes:

- Classification of rows on each page into input sections
- Input file page properties
- Grouping of pages as sets
- Mapping of the contents of input sections as data variables
- Definition of derived variables

What would you like to know about?

- Input Sections
- Examining how Sections are Populated
- Design Principles
- Setting the Input Page Size
- Defining Sets
- Working with Data Variables
- Common Input Section Tasks
- Working with Input Header Sections
- Working with Detail Sections
- Working with Input Footer Sections

To create the text sample project

1. To start the wizard, click the New Project button on the Main toolbar.
2. From the Type drop-down, select Text from the available file types.
3. Click to display the Select Input File dialog and select the file Chapter5Sample.txt from the <install dir>\CF6Samples\Tutorials folder.
4. Click Next.
5. Select the Create a new DataMap radio button and click Next. A new DataMap will be created.

6. Select the Standard radio button and click Next. This option will permit you to fully define the structure of the input file.

7. Click **Finish**. The new sample project will be displayed in the Input Window.

![Input Window showing the sample project input file](image-url)

For more information about screen elements in the Input Window, refer to **Understanding the Screen Elements**.
Input Sections

There are four types of sections used to define the structure of the input file:

- **Input file header section**
  (Refer to Working with the Input File Header Section for more information.) A non-repeating section at the start of the file that contains header text.

- **Input header section**
  (Refer to Working with the Input File Header Section for more information.) A section at the top of each page that contains header text.

- **Detail section**
  (Refer to Working with Detail Sections for more information.) Used to define repeating rows within the body of the page.

- **Input footer section**
  (Refer to Working with Input Footer Sections for more information.) A section at the bottom of each page that contains footer text.

The following graphic displays the way you would define the sections of the sample file that are repeated on each page of the input file. The input file header section occurs once, at the start of the file.

![Figure 5.5: Input Page Structure](image-url)
Each section in the input file has a different set of properties. Refer to Display and Edit Input Section Properties for more information.

Examining how Sections are Populated

To assist you in the design process, you can examine all the instances of the same section in the input file with the Input>Show All Instances command. In this view mode, all non-blank characters in all instances of the selected section will be marked with an “X”.

![Populated column positions are marked with an X](image)

*Figure 5.6: Examining all instances of a section in the input file*

Design Principles

Create!form Director can handle a variety of input file structures, from simple tables to complex and irregular forms. This section introduces the design principles that form the foundation for the procedures described later in this chapter.

Designing from the Top-Down

The most important principle, when preparing the input file, is to approach the design from the top-down. As depicted in the Design flowchart, start with the input header section on the first page and make any adjustments needed to the section before looking at the next section. Scan the page from the top-down and locate the first anomaly. Resolve the anomaly, then move down the page.
to the next anomaly. When the design is correct on the first page, view the second page, again from the top-down. Proceed through the file until the design is complete from start to finish — from top to bottom.

**Design Flowchart**

The following flowchart describes the main tasks associated with input design.
Designing for Text Input Files

Figure 5.7: Input design flowchart
Setting the Input Page Size

Create!form Director can handle fixed or variable page lengths, and widths of up to 2048 characters. You can define the input file page length in the following ways:

- By a form feed character (default setting)
- The page has a fixed number of lines
- By a page number
- By a specified character or character string

The input page size is set from the Input File Properties dialog:

![Input File Properties dialog](image)

- Sets the input page length.
- Sets how the end-of-page or start-of-page is identified.
- Only required when a custom end-of-file character is used in the input file.
- Sets the input page width.
- Scans the input file to determine page length and width. (Refer to How the Page Size is Calculated for more information.)

For information on other input file properties refer to Other Input File Settings.

*Figure 5.8: Input page size options on the Input File Properties dialog*
To define the input file page width and length

1. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.

2. In the Lines field, type the page length in lines.

3. In the Columns field, type the line width in characters. If you are unsure of the page length or line width, click the Recalculate button to scan the input file and find the maximum page length and line width in the file. For more information refer to How the Page Size is Calculated.

4. In the End of page section, select one of the following radio button options:
   - **Form feed**
     A new page will be generated whenever a form feed character is encountered.
   - **Fixed number of lines**
     A new page will be generated according to the Lines field above.
   - **Characters**
     A new page will be generated when the specified characters or character string is encountered.
     In the Characters field, type the character or character string.
     If the string is in a specific location, type the starting position of the string in the at column field; all other occurrences of the string will be ignored.
     Select the appropriate radio button to determine how the character string will be applied:
     - Select define the first line of a page if the new page begins at the start of the line containing the string;
     - Select define the last line of the page if the new page begins at the start of the next line following the line containing the string;
     - Select terminate the page if the new page begins at the next character following the string.

5. Click OK.
How the Page Size is Calculated

Once Create!form Director has calculated the page size for a project, it remains unchanged until the Page size fields are edited, or the Recalculate button is selected.

When creating a new project
When creating a new standard or simple project, Create!form Director will read the first 10 pages of the file to determine the maximum lines and columns per page. If the number of lines is less than the current default value, the default will be used. If the number of columns is less than the current default value, the default will be used.

When recalculating the page size
When you click the Recalculate button, Create!form Director will read the first 10 pages of the file to determine the maximum lines and columns per page. The calculated page size is displayed and the default values are ignored.

To change the default number of lines and columns
1. In the Lines and Columns fields, type the new default values.
2. Review the other settings in the Input File Properties dialog and set default values as appropriate.
3. Click the Set as Default button.

The default settings may be applied when a new project is created (see above) or whenever the Reset from Default button is selected.

Other Input File Settings

In addition to page size settings, the following properties are also set from the Input File Properties dialog:

- **Data format**
  Sets the formats used by data variables (Refer to Set Input Data Formats for more information).

- **Ignore leading blank pages**
  Select to ignore blank pages at the start of the input file.

- **File starts with document header**
Designing for Text Input Files

- **Ignore bold and underline formatting**
  
  When processing OS/400 spool files, if the spool file contains bold and underline overstrike control characters, you can choose to either ignore overstriking or convert the overstrike characters into PostScript. Previously this option could only be set in the writer properties on the server.

---

**Figure 5.9: The input File Properties dialog**

- Ignores blank pages at start of input file.
- Ignores overstrike characters in OS/400 spool files.
- Sets the formats used by data variables.
- Creates an input file header section.

For information on page size settings, refer to Setting the Input Page Size.
To define input file properties

1. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.

2. Select options as required.

3. Click OK.
Defining Sets

You can use the following means to determine when a new set starts:

- The entire input file can be treated as one set (default setting)
- When the value in one or more columns changes
- Or, one set per row

In addition, you can now add conditional settings for each new set, using three options to define Child Tag values.
To define sets

1. From the Main menu, select Set > Define.
2. On the Define Set screen, select a desired tag:
   - Treat entire spool file as one set.
• When values change in marked columns:
  Select one or more columns.
• One set per row

3 (optional) In the Conditional set logic section, you can also configure additional settings contingent on Child Tags values.
  • In the Condition set logic section, click Conditionally define sets to activate the Child Tags field box.
  • Highlight the Child Tag you want to work with and select one of the desired radio button options. (For example, if you are working with an Invoice set, you might want to select invoice_number and click the “When value changes” radio button).

4 Click OK.

## Working with Data Variables

### Set Input Data Formats

Before you start creating data variables, you must set the data formats that Create!form Director will use to recognize dates and numeric data in the input file. For example, you can change the decimal separator to a comma, or the date order from year-month-day to month-day-year.

**To change the input data format**

1 From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.

2 From the Date order drop-down, select the required date format.

3 From the Decimal symbol drop-down, select the required numeric format.

4 Click OK.
Create Data Variables

To selectively map particular data from the input, for use in the project, you must create data variables around the data.

To create a data variable

1. In the Input Window, select the section containing the data you want.
2. Click-and-drag a box around the area that contains the information you want to store in the variable. The Data Variable Properties dialog will be displayed.

3. In the Name field, type a name for the data variable. The variable name must start with an alphabetic character and then be followed by any combination of alphabetic (a-z, A-Z) and numeric (0-9) characters and underscores. Names are not case sensitive. Naming of variables is entirely at the discretion of the user; however, the convention of naming variables to represent the data they contain is recommended. You should think of a name that you will be able to recognize and distinguish from all other data variables in the section.
4. Select a data type from the Type drop-down list.
5. Click OK.

The area defining the variable will appear highlighted.
Note

When sizing a data variable, use the Show All Instances command to examine the extents of the data in all instances of the section in the input file.

To change the size and position of a data variable
1 In the Input Window, select the data variable.
2 Click-and-drag the resize handles to change the box size, or click-and-drag inside the box to reposition the box.

To display and edit data variable properties
1 Display the Data Variable Properties dialog by doing one of the following:
   • Double-click the data variable in either the Input Window or the Input Tree.
   • Right-click the data variable in either the Input Window or the Input Tree and select Properties from the shortcut menu.
2 With the Data Variable Properties dialog you can:
   • Change the name of the data variable by typing a new name in the Name field
• Select the data type from the Type drop-down (refer to Set Input Data Formats for more information)

• Change the position and size of the data variable in the Row, Column, Depth and Length fields
  Row and column positions are defined relative to the origin (top left corner) of the selected section

• Select the Trim spaces checkbox to remove leading and trailing spaces from the text string

• Select the Expandable checkbox to make the data variable expand with the section (refer to Make Data Variables Expandable for more information)

To delete a data variable
1  Select the data variable in the Input Window or Input Tree.
2  Press the Del key.

Create Data Variables in a Detail Section

Any data variables created in one occurrence of the detail section will be applied automatically to all other occurrences of the same detail section.

Refer to Create Data Variables for more information.

The area defining the variable will appear highlighted and will be displayed in each occurrence of the same detail section:
Make Data Variables Expandable

Data variables defined in expandable sections can be made to expand as the section expands (Refer to Make an Input Header Section Expandable or Make a Detail Section Expandable for more information.) For example, you could define an expandable data variable on the first row of an expandable detail section so that when the section expands to two or more rows, the data variable will contain the extra rows. An expandable data variable will always expand to the bottom of the section.

To define an expandable data variable

1. Create a data variable in an unexpanded occurrence of an expandable input section. The Data Variable Properties dialog will be displayed.
2. Type the data variable name in the Name field, and edit other properties as required.

---

**Figure 5.12: Creating a data variable in a detail section**

<table>
<thead>
<tr>
<th>Data Variable Properties dialog displayed.</th>
<th>Selected Detail Section.</th>
<th>Repeated occurrences of the same detail section.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data variable in an unexpanded occurrence of an expandable input section.</td>
<td>Data variable is displayed in all other occurrences.</td>
<td></td>
</tr>
</tbody>
</table>
3 Select the Expandable checkbox.
4 Click OK.

Common Input Section Tasks

The following tasks are applicable to all input sections (header, detail and footer).

**What do you want to know about?**
- Display and Edit Input Section Properties
- To control when an input section is generated
- Working with Multiple Input Sections
- Deleting Input Sections

For tasks specific to each section type refer to:
- Working with the Input File Header Section
- Working with Input Header Sections
- Working with Detail Sections
- Working with Input Footer Sections

Display and Edit Input Section Properties

Input section properties can be displayed so that you can:
- Change the name of the section
- Change the size of the section
- Control where this type of section is applied in the input file
- Control when this type of section is generated
- Associate which types of sections can follow this section
- Control the order in which following sections are tested

When you change the properties of a section, the changes are applied to all occurrences of that section in the file.
To display the Input Section Properties dialog

Do one of the following:

- With the section active, select Input > Section Properties from the Main menu.
- Double-click the selected section.
- Right-click the selected section and select Section Properties from the shortcut menu.
- Double-click the section tab on the Status bar.
Input Section Properties Dialog Controls

For examples of how to use conditions refer to Add Input Header Sections, Add Detail Sections, and Add Input Footer Sections.

Refer to Chapter 9, Expressions and Conditions, for information on how to use the Condition Builder.

Refer to Make an Input Header Section Expandable and Make a Detail Section Expandable for information on creating expandable sections.

Refer to To control when an input section is generated.

For information on controlling when a section is generated refer to To control when an input section is generated.

Use these controls to add or remove sections from the Followed By list.

Use these controls to change the order in which sections are tested in the Followed By list.

The followed By list. For information on how to use the Followed By list, refer to Working with Multiple Input Sections.

Click to display the sections available to be added to the Followed By list.

Figure 5.13: The input section properties dialog
Controlling When an Input Section is Generated

You can use the Generate Section options in the Input Section Properties dialog to control when an input section is generated in the project. If the section is not generated, the position on the input page where the section would have been displayed is reserved. Changing the Generate Section options will not alter the sequence in which sections are applied to the input page.

To control when an input section is generated

1. In the Input Window, double-click the section. The Input Section Properties dialog will be displayed.

2. Select one of the radio buttons in the Generate Section field:
   - **Always** — the section will always be generated if valid.
   - **Never** — the section will never be generated even if valid.
   - **If not empty** — the section will only be generated if it contains printable characters.

Working with Multiple Input Sections

In cases where input pages begin with different header text, or end with different footer text, or contain different line items, you can define and use additional input sections that accommodate these anomalies. Note that:

- All input header sections are listed in the input headers control list
- All detail sections are listed in the Followed By list on the relevant Input Section Properties dialog
- All input footer sections are listed in the input footers control list

For each page, the logic of which section appears where, is summarized in the following flowchart:
Deleting Input Sections

You can delete the selected input section with the **Input > Delete Section** command. If the section is not visible on the current page, select the section in the Input Tree. You should exercise great care when deleting an input section, as this can disrupt the logic and continuity of how other sections are applied.

Working with the Input File Header Section

You can create an input file header section to contain non-repeating data at the start of the input file.

**To create an input file header section**

1. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.
2. Select File starts with document header checkbox.
3. Click **OK**.
To adjust the input file header section size
1. Select the input file header section.
2. Click-and-drag the resize handle on the bottom of the section to reduce or increase the section size.

You can also double-click the section in the Input Window and set the number of rows from the Input Section Properties dialog.

Working with Input Header Sections

What do you want to do?
- Adjust the Input Header Section Size
- Make an Input Header Section Expandable
- Add Input Header Sections

What do you want to know about?
- Rules for Creating Multiple Input Header Sections

Adjust the Input Header Section Size

Typically when you start a new project, the default size of the input header section will be too big or too small for the header text in the input file. You must adjust the input header section so that it contains all the header text and only the header text.
Note

Use the Show All Instances command to examine all instances of the section in the input file.

To adjust the input header section size

1 Select the input header section.

2 Click-and-drag the resize handle on the bottom of the section to reduce or increase the section size.

Make an Input Header Section Expandable

By default, input header sections have a set size. That is, all instances of that section will have the same size regardless of their content. Where the size of the text in an input header section varies between instances, you can make the section dynamically expand around the relevant text. The section becomes “expandable”.

To make an input header section expandable

1 In the Input Window, double-click the input header section. The Input Section Properties dialog will be displayed.

2 In the Size field, type the minimum size of the section in rows.

3 Select the Expandable checkbox.

4 Click OK.

The section size will be reset to the minimum size. Select the first detail section below the input header section and identify which data variable can be used to test when this detail section is valid. The data variable will be used to build a condition that tests TRUE in the detail section and FALSE in the expanded rows of the input header section.

5 Double-click the first detail section below the input header section. The Input Section Properties dialog will be displayed.

6 Click the button next to the Condition field. Use the Condition Builder to create a valid condition, using the identified variable, for this detail section.

7 Click OK. The input header section will now automatically expand in size, testing each row below the section, until the condition in a following detail section tests TRUE.
8 Ensure that all sections that are permitted to follow the header section, contain a suitable condition that will distinguish header rows from detail rows.

The sections that are permitted to follow the input header section are listed in the This section can be followed by field on the input header Input Section Properties dialog — also referred to as the Followed By list. The input header section will not expand if any of the sections in the Followed By list do not contain a condition, or the condition always tests TRUE.
Note
You can also make data variables expandable in an expandable section.

Add Input Header Sections

In cases where the content of the input header section changes between pages, or an expandable header section cannot be used, you can add additional header sections. Only one input header section can be used on a page.

To create an additional input header section

1. From the Main menu, select Input > Input Headers. The Input Headers Control List dialog will be displayed.

2. Click the Add Row button. The New Header Section dialog will be displayed.

3. In the Name field, type the new input header section name.

4. In the Size field, type the number of rows required.

5. Using the controls provided, add one or more detail sections to the This section can be followed by list. If there are no sections in the Followed By list, or if the sections listed are not valid when tested, no detail sections will be created on the page.

6. Click OK to close the New Header Section dialog. Do not close the Input Headers Control List dialog. The input headers control list will now have two input header sections defined. Both input header sections are unconditional (the Condition field is empty on the Input Section Properties dialog). If the first input header in the list is unconditional, it will always be created. So the first input header must be conditional. If the condition tests TRUE then the first input header is applied, if the test fails then the second input header is applied.

7. Click the Edit button next to the first input header in the control list. The Input Section Properties dialog will be displayed.

8. Click the button next to the Condition field. Use the Condition Builder to create a valid condition for this input header section. If you wish to use a data variable in the condition, you must create the data variable first.

9. Check that the detail sections in the This section can be followed by list are valid and in the correct order.
10 Click OK to close the Input Section Properties dialog.

11 Click the Edit button on the second (unconditional) input header to display the Input Section Properties dialog.

12 Check that the detail sections in the This section can be followed by list are valid and in the correct order.

13 Click OK to close the Input Section Properties dialog.

14 Click OK to close the Input Headers Control List dialog.

The definition of the new input header section will be applied to the entire file.

---

**Note**

The Condition Builder makes available a number of useful system variables and conditions, for example the condition Sys.FirstPageOfDoc will test TRUE on the first page of the document.

---

**To delete an input header section**

Care should be taken whenever a input header section is deleted as this will alter not only which input header section is applied, but may also change the way detail sections are applied to the rest of the page. Always review the Followed By lists on the other input header sections, and the order of the input headers control list, before deleting a input header section.

1 From the Main menu, select Input > Input Headers. The Input Headers Control List dialog will be displayed.

2 Select the input header section in the control list.

3 Click the Delete Row button.

4 Confirm that you wish to delete this section.

5 Click OK.

The change will be applied to the entire file.
INPUT DESIGN

Designing for Text Input Files

Rules for Creating Multiple Input Header Sections

The procedures can be applied when adding further input header sections. The rules applying to multiple input header sections are:

- The last header in the input headers control list should be unconditional.
- All other headers in the list should be conditional.
- The first header that tests TRUE in the list is applied.
- If none of the headers in the control list test TRUE on a page, then no header will be created on that page.
- Input header sections can be expandable.

The detail section that follows the input header section, will be the first valid detail section in the Followed By list of the input header section.

Working with Detail Sections

What do you want to do?

- Add Detail Sections
- Make a Detail Section Expandable

What do you want to know about?

- About Defining Detail Sections
- Rules for Creating Multiple Detail Sections

About Defining Detail Sections

You can create additional detail sections when the content of line items varies within the page. For example, an invoice may contain subtotal lines, blank lines or description lines in addition to the regular line items. Three different detail section types are shown in the following example.
The details of lines are as follows:

- Type 1 is the primary line item containing the item details, quantity and price
- Type 2 is a descriptive line relating to the previous type 1 line
- Type 3 is a subtotal line

In particular note that:

- Type 1 lines are not always followed by type 2 lines
- Type 2 lines can only follow type 1 lines
- Type 2 lines always contain “***” at the start of each line
- Type 2 can be one or more lines
- Type 3 is always two lines
- Type 3 lines can follow either type 1 or 2 lines
- Type 3 lines can occur at any position
- Type 3 lines always contain the text string “Subtotal”

These observations form a set of rules that define the input file structure. The process of building a DataMap involves translating these observations or rules into section properties. You must create a new section for each of the line types you observe in the file.

**Figure 5.15: An example of irregular line items**

<table>
<thead>
<tr>
<th>Type</th>
<th>Description</th>
<th>Quantity</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Type 1</td>
<td>APT103 Bamboo Blinds</td>
<td>10</td>
<td>1020.00</td>
</tr>
<tr>
<td>Type 1</td>
<td>APT103 Black Dragon Vase</td>
<td>10</td>
<td>1990.00</td>
</tr>
<tr>
<td>Type 2</td>
<td>*** 9 units shipped, balance with next order</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Type 2</td>
<td>*** hold for collection at Nunawading warehouse</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Type 3</td>
<td>Subtotal</td>
<td>12590.00</td>
<td></td>
</tr>
<tr>
<td>Type 1</td>
<td>APT103 Tea Chests</td>
<td>4</td>
<td>3196.00</td>
</tr>
<tr>
<td>Type 1</td>
<td>APT103 Black Dragon Vase</td>
<td>10</td>
<td>1990.00</td>
</tr>
<tr>
<td>Type 2</td>
<td>***</td>
<td>***</td>
<td>***</td>
</tr>
<tr>
<td>Type 1</td>
<td>AT1C78 Cane Baskets</td>
<td>5</td>
<td>250.00</td>
</tr>
<tr>
<td>Type 1</td>
<td>AT1C38 Black Dragon Set</td>
<td>1</td>
<td>1600.00</td>
</tr>
<tr>
<td>Type 3</td>
<td>Subtotal</td>
<td>7036.00</td>
<td></td>
</tr>
</tbody>
</table>
Add Detail Sections

There are many possible ways of creating DataMaps for the same input file. The type of sections defined will depend not only on the content of the input file, but also on how the input data will be used in the project.

To define additional detail sections

1. Identify the first instance where a new section is required.
2. Select the section immediately before (above) where the new section is required.
3. From the Main menu, select Input > New Section After. The New Data Section dialog will be displayed.
4. In the Name field, type the name of the new section.
5. In the Size field, type the size of the section in rows.
6. Click OK. The new section will be displayed in the Input Window.
7. Select the new section.
8. Create the data variables required to define the data in the new section.
9. Create any additional data variables that are required to assist in distinguishing this section from all other detail sections. You only need to create additional data variables if the data variables created in the previous step are not suitable.

For example, you might test when a particular character position is blank, or when a character string contains a particular substring, or a combination of tests using boolean operators.

10. From the Main menu, select Input > Section Properties. The Input Section Properties dialog will be displayed.
11. Click the button to display the Condition Builder dialog and create a condition that tests TRUE when this section is valid.

Note

Use the Show All Instances command to examine all instances of the section in the input file.
12 Define the sections that can follow this section in the This section can be followed by field by adding, removing and ordering sections with the controls provided. The order in which the sections appear in this list is very important: the first valid section in this list will be the next section created after this section.

13 Click OK.

The definition of the new section will be applied to the entire file and displayed in the Input Window.

---

**Note**

*It is good design practice to create a positive condition for every detail section you create, even if the condition is not immediately required to identify the section.*

---

**Make a Detail Section Expandable**

In cases where the number of rows in a detail section is variable, e.g. repeated comment lines, you can make the detail section expandable.

**To make a detail section expandable**

1 In the Input Window, double-click the detail section. The Input Section Properties dialog will be displayed.

2 In the Size field, type the minimum size of the section in rows.

3 Select the Expandable checkbox.

4 Click **OK**.

The section will now automatically expand in size, testing each row below the section, until a condition in a following detail section tests TRUE. Ensure that all sections that are permitted to follow the section contain a suitable condition.
Note
You can also make data variables expandable in an expandable section.

Rules for Creating Multiple Detail Sections

The rules applying to multiple detail sections are:

- The detail section that follows the input header section, will be the first valid detail section in the Followed By list of the input header section.
- The first valid detail section in the previous detail section Followed By list is applied next.
- An unconditional detail section will always be applied if tested in the previous detail section Followed By list.
- If there are no valid sections in the last Followed By list, no further detail sections will be applied to that page.
- Detail sections can be expandable.

Working with Input Footer Sections

What do you want to do?

- Adjust the Input Footer Section Size
- Add Input Footer Sections

What do you want to know about?

- Rules for Creating Multiple Input Footer Sections

Adjust the Input Footer Section Size

As with the input header section, you must adjust the size of the input footer section so that it contains all footer text and no text that belongs in a detail section.
Note

*Use the Show All Instances command to examine all instances of the section in the input file.*

---

**To adjust the input footer section size**

1. Select the input footer tab on the Status Bar to select and display the input footer section.

2. Click-and-drag the resize handle on the top border of the input footer section to make the section bigger or smaller.

**Add Input Footer Sections**

In cases where the content of the input footer section changes between pages, you can add and use additional input footer sections. The additional input footer sections will be added to the input footers control list. Only one can be used on each page.

**To define additional input footer sections**

1. From the Main menu, select Input > Input Footers. The Input Footers Control List dialog will be displayed.

2. Click the Add Row button to display the New Footer Section dialog.

3. In the Name field, type the footer section name.

4. In the Size field, type the number or rows required for the new input footer section.

5. Click OK to close the New Footer Section dialog. Do not close the Input Footers Control List dialog.

The input footers control list now has two input footer sections defined. Both input footer sections are currently unconditional (the Condition field is empty). If the first in the list is unconditional, it will always be created. So the first footer must be conditional. If the condition tests TRUE then the first input footer is applied, if the test fails then the second input footer is applied.

6. Click the Edit button for the first input footer to display the Input Section Properties dialog.
7 In the Condition section, click the button to display the Condition Builder dialog and create a condition that tests TRUE when this section is valid. For information on creating conditions, see Chapter 9.

8 Click OK to close the Input Section Properties dialog.

9 Click OK to close the Input Footers Control List dialog.

The definition of the new input footer section will be applied to the entire file.

---

Note

The Condition Builder makes available a number of useful system variables and conditions, for example the condition Sys.LastPageOfSet will test TRUE on the last page of each set.

---

Rules for Creating Multiple Input Footer Sections

The procedures can be applied when adding further input footer sections. The rules applying to multiple input footer sections are:

- The first input footer that tests TRUE in the control list is applied.
- If none of the input footers in the control list test TRUE on a page, then no input footer will be created on that page.
Designing for XML Input Files

The input design process for XML files involves identifying the different elements of the XML input file; this information is recorded in the DataMap. A DataMap for an XML input file is denoted by the .xmp file name extension.

What would you like to know about?

• XML File Structure
• Input File Properties
• Defining Sets
• Defining Sections
• Working with Data Variables

To create the XML sample project

1. To start the wizard, click the New Project button on the Main toolbar.
2. From the Type drop-down, select XML from the available file types.
3. Click to display the Select Input File dialog and select the file Chapter5Sample.xml from the <install dir>\CF6Samples\Tutorials folder.
4. Click Next.

Select the Create a new DataMap radio button and click Next. A new DataMap will be created.

5. Click Finish. The new sample project will be displayed in the Input Window.

XML File Structure

To be properly interpreted, the XML file should be “well-formed” and adhere to the W3C standards published by World Wide Web Consortium. An example of a well-formed XML file is shown below:
Table 6.1:

<table>
<thead>
<tr>
<th>XML Element Type</th>
<th>Line</th>
<th>XML Source</th>
</tr>
</thead>
<tbody>
<tr>
<td>XML declaration</td>
<td>1</td>
<td>&lt;?xml version=&quot;1.0&quot; encoding=&quot;ISO8859-1&quot; ?&gt;</td>
</tr>
<tr>
<td>comment</td>
<td>2</td>
<td>&lt;!-- Ghoul Club Member Updates--&gt;</td>
</tr>
<tr>
<td>root tag with attribute</td>
<td>3</td>
<td>&lt;update total=&quot;2&quot;&gt;</td>
</tr>
<tr>
<td>child element start tag with attribute</td>
<td>4</td>
<td>&lt;member count=&quot;1&quot;&gt;</td>
</tr>
<tr>
<td>subchild element tag</td>
<td>5</td>
<td>&lt;ID&gt;1984&lt;/ID&gt;</td>
</tr>
<tr>
<td>subchild element start tag</td>
<td>6</td>
<td>&lt;changes&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>7</td>
<td>&lt;street&gt;1667 Elm Street&lt;/street&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>8</td>
<td>&lt;email&gt;<a href="mailto:fred.krueger@craven.com">fred.krueger@craven.com</a>&lt;/email&gt;</td>
</tr>
<tr>
<td>subchild element end tag</td>
<td>9</td>
<td>&lt;/changes&gt;</td>
</tr>
<tr>
<td>child element end tag</td>
<td>10</td>
<td>&lt;/member&gt;</td>
</tr>
<tr>
<td>child element start tag with attribute</td>
<td>11</td>
<td>&lt;member count=&quot;2&quot;&gt;</td>
</tr>
<tr>
<td>subchild element tag</td>
<td>12</td>
<td>&lt;ID&gt;1886&lt;/ID&gt;</td>
</tr>
<tr>
<td>subchild element start tag</td>
<td>13</td>
<td>&lt;changes&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>14</td>
<td>&lt;title&gt;Mr&lt;/title&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>15</td>
<td>&lt;first&gt;Edward&lt;/first&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>16</td>
<td>&lt;last&gt;Hyde&lt;/last&gt;</td>
</tr>
<tr>
<td>nested subchild element tag</td>
<td>17</td>
<td>&lt;email&gt;<a href="mailto:henry.jekyll@stevenson.com">henry.jekyll@stevenson.com</a>&lt;/email&gt;</td>
</tr>
<tr>
<td>subchild element end tag</td>
<td>18</td>
<td>&lt;/changes&gt;</td>
</tr>
<tr>
<td>child element end tag</td>
<td>19</td>
<td>&lt;/member&gt;</td>
</tr>
<tr>
<td>end root tag</td>
<td>20</td>
<td>&lt;/update&gt;</td>
</tr>
</tbody>
</table>

Figure 5.16: Example of a well-formed XML file

The elements and structure of the XML file are interpreted according to an underlying set of rules that determine how the XML elements will be applied in the output project. These initial settings can be changed with the following commands:
### Input File Properties

The input file properties determine the input format for dates and numbers, and which tags will be ignored when the input file is processed.

**To change the input file properties**

1. Select the Input Window by clicking anywhere inside the input window frame.

2. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.

<table>
<thead>
<tr>
<th>To do this</th>
<th>Select</th>
</tr>
</thead>
<tbody>
<tr>
<td>Change formats of date and numeric data</td>
<td>Input &gt; File Properties (refer to Input File Properties for more information)</td>
</tr>
<tr>
<td>Ignore selected XML tags in source file</td>
<td>Input &gt; File Properties (refer to Input File Properties for more information)</td>
</tr>
<tr>
<td>Change how sets are created</td>
<td>Project &gt; Define Set (refer to Defining Sets for more information)</td>
</tr>
<tr>
<td>Change how sections are created</td>
<td>Input &gt; Tag Properties (refer to Defining Sections for more information)</td>
</tr>
<tr>
<td>Rename a section</td>
<td>Input &gt; Tag Properties (Refer to Defining Sections for more information)</td>
</tr>
</tbody>
</table>
In the Data format section, select the format used for date and decimal values in the input file.

In the Ignored Tags section, use the toolbar buttons to add the tags you want to ignore or remove the tags you want to include.

Click **OK**.

The changes will be applied to the input file and the input window will be refreshed.

**Defining Sets**

You can use the following means to determine when a new set starts:

- The entire input file can be treated as one set (default setting)
- When the value contains a certain value
- When the value contains a certain character

In addition, you can now add conditional settings for each new set, using three options to define Child Tag values.
To define sets

1. From the Main menu, select Set > Define.
2. On the Define Set screen, select a desired tag:
   - Treat entire spool file as one set.
   - When values change in marked columns:
     - Select one or more columns.
   - When the value contains a certain character
3 (optional) In the Conditional set logic section, you can also configure additional settings contingent on Child Tags values.
   • In the Condition set logic section, click Conditionally define sets to activate the Child Tags field box.
   • Highlight the Child Tag you want to work with and select one of the desired radio button options. (For example, if you are working with an Invoice set, you might want to select invoice_number and click the “When value changes” radio button).

4 Click OK.

Defining Sections

You can select one or more tags to indicate where a new data section will be created:

To define sections
1 In the Input Window, select a tag that defines the start of a new section.
2 From the Main menu, select Input > Tag Properties.
3 The Tag Properties dialog will be displayed.
4 Select the Create section checkbox and type the section name.
5 Click OK.

Working with Data Variables

Data variables are created automatically from the tag attributes and data elements in the input file. The data variables created are shown in the Input Tree view:
Designing for XML Input Files

What would you like to know about?

- Changing the Properties of Data Variables
- Changing the Input Data Format

Changing the Properties of Data Variables

To change the properties of data variables

1. In the Input Window, double-click the data variable. The Data Variable Properties dialog will be displayed.
With the Data Variable Properties dialog you can:

- Change the name of the data variable by typing a new name in the Variable name field.
- Select the data type from the Data type drop-down.

Click OK.

![Data Variable Properties dialog](image)

**Figure 5.19: Data variable Properties dialog**

### Changing the Input Data Format

Create!form Director interprets dates and numeric data in the input file using the data formats defined by the input file properties. For example, you can change the decimal separator to a comma, or the date order from year-month-day to month-day-year.

**To change the input data format**

1. From the Main menu, select Input > File Properties. The Input File Properties dialog will be displayed.
2. From the Date order drop-down, select the required date format.
3. From the Decimal symbol drop-down, select the required numeric format.
4. Click **OK**.
Designing for JDE PDF Input Files

The J.D. Edwards PDF format is a self-defining proprietary PDF format generated by PeopleSoft EnterpriseOne. The contents of the PDF file are pre-mapped, which Create!form Director automatically translates into sections and data variables.

The input design process for JDE PDF files involves defining how sets are created and defining any derived variables that are required in the project; this information is recorded in the DataMap. A DataMap for an JDE PDF input file is denoted by the .jmp6 file name extension.

To view the input file elements mapped by the DataMap
Click the Input Tree on the Main toolbar or select View > Input Tree from the Main menu.

What would you like to know about?

• Input File Properties
• Defining Sets
• Input Sections
• Data Variables
Input File Properties

Use the Input > File Properties. command to change the following input file properties:

- **Non-breakable sections**
  When the file is repaginated, any input section defined as a non-breakable section will not be permitted to span a page break.

- **Date order**
  Select the date order format used in the input file.

- **Can contain duplicates**
  If the input file can contain duplicates, this option forces a scan for EOF markers and records the number of duplicates in the Sys.PDFNumCopies system variable.

**To add and remove non-breakable input sections**

1. Select the Input Window by clicking anywhere inside the input window frame.
2. From the Main menu, select Input > File Properties.
3. To add section to the Non-breakable sections, click the Add Row button, click the drop-down button on the new row, and select the section from the list.
4. To remove a section, select the section in the Non-breakable sections list, and click the Delete Row button.
5. Click OK.

**To change the input file date format**

1. Select the Input Window by clicking anywhere inside the input window frame.
2. From the Main menu, select Input > File Properties.
3. Select the required format from the Date order field.
4. Click OK.
Defining Sets

You can use the following means to determine when a new set starts:

- The entire input file can be treated as one set (default setting)
- When the value in one or more columns changes
- Or, one set per row

In addition, you can now add conditional settings for each new set, using three options to define Child Tag values.
To define sets
1. From the Main menu, select Set > Define.
2. On the Define Set screen, select a desired tag:
   - Treat entire spool file as one set
- **When values change in marked columns:**
  Select one or more columns
- **One set per row**

3 (optional) In the Conditional set logic section, you can also configure additional settings contingent on Child Tags values.

- In the Condition set logic section, click Conditionally define sets to activate the Child Tags field box.
- Highlight the Child Tag you want to work with and select one of the desired radio button options. (For example, if you are working with an Invoice set, you might want to select invoice_number and click the “When value changes” radio button).

4 Click **OK**.

**Input Sections**

Input sections are created automatically from the pre-mapped contents of the input file. The sections defined in the input file are shown in the Input Tree view. The properties of input sections in a JDE PDF input file cannot be edited.

**Data Variables**

Data variables are created automatically from the pre-mapped contents of the input file. The data variables are shown in the Input Tree view. The properties of data variables in a JDE PDF input file cannot be edited.

Refer to **Creating Derived Variables** for more information on creating your own variables from expressions containing data variables and other types of information.
Creating Derived Variables

Derived variables are expressions that are used to combine and manipulate the values of data variables. To create a derived variable, select Input>Derived Variables from the Main menu to display the Derived Variables dialog. Refer to Expressions for more information.

Examples

<table>
<thead>
<tr>
<th>Derived Variable</th>
<th>Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>TotalPrice</td>
<td>(UnitPrice+SalesTax)*Qty</td>
</tr>
<tr>
<td>LastChar</td>
<td>substring(ProdDesc,length(ProdDesc),1)</td>
</tr>
<tr>
<td>AddressLine1</td>
<td>Title+FirstName+LastName</td>
</tr>
<tr>
<td>DaysInMonth</td>
<td>Day(date(year(Sys.Today),month(Sys.Today)+1,1)-1)</td>
</tr>
</tbody>
</table>

Refer to Data Variables, Derived Variables and Section Variables for more information on how to use derived variables in a project.
Nodes define the actions you want the server to perform on the input file and the destinations it will be sent to.

The following topics are covered in this chapter:

- Key Design Concepts
- Adding Nodes
- Selecting and Editing Nodes
- Node Properties
- Window Display Preferences
Key Design Concepts

A Create!form Director project defines the sequence and logic for processing, merging and delivering input files. The procedures for designing a project involve creating a flowchart constructed from nodes and connectors. Each node represents either an action, a destination or a decision, which is applied to the input file. The project is created and edited graphically in the Design Window. In the sample project shown below, the data in the input file is transformed, merged, faxed, emailed, printed, archived and extracted:
Figure 6.1: Sample design
What do you want to know about?

- About Nodes
- About Parent, Sibling and Child Relationships
- Sequence and Timing
- Using Variables and Expressions to Control Actions

About Nodes

There are three types of nodes:

- Destination - sends data to a destination
- Decision - redirects data by applying a condition
- Action - performs an action on the data

The nodes available in a Create!form Director project are summarized below:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Send to Current Device" /></td>
<td>Send to Current Device</td>
<td>Destination</td>
<td>Sends data to the current device</td>
</tr>
<tr>
<td><img src="image" alt="Send to Queue" /></td>
<td>Send to Queue</td>
<td>Destination</td>
<td>Sends data to a print queue</td>
</tr>
<tr>
<td><img src="image" alt="Send to Folder" /></td>
<td>Send to Folder</td>
<td>Destination</td>
<td>Sends data to a network folder</td>
</tr>
<tr>
<td><img src="image" alt="Send to FTP" /></td>
<td>Send to FTP</td>
<td>Destination</td>
<td>Sends data to an FTP address</td>
</tr>
</tbody>
</table>
### Table 6.1:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Send to HTTP Icon" /></td>
<td>Send to HTTP</td>
<td>Destination</td>
<td>Sends data to an HTTP address</td>
</tr>
<tr>
<td><img src="image" alt="Condition Icon" /></td>
<td>Condition</td>
<td>Decision</td>
<td>Redirects by applying a condition</td>
</tr>
<tr>
<td><img src="image" alt="Split Icon" /></td>
<td>Split</td>
<td>Action</td>
<td>Partitions the input file into subdocuments</td>
</tr>
<tr>
<td><img src="image" alt="Sort Icon" /></td>
<td>Sort</td>
<td>Action</td>
<td>Sorts input file partitions</td>
</tr>
<tr>
<td><img src="image" alt="Sub Project Icon" /></td>
<td>Sub Project</td>
<td>Action</td>
<td>Processes data through another Create!form Director project</td>
</tr>
<tr>
<td><img src="image" alt="Merge Icon" /></td>
<td>Merge</td>
<td>Action</td>
<td>Merges data with a Create!form Designer project</td>
</tr>
<tr>
<td><img src="image" alt="Transform Icon" /></td>
<td>Transform</td>
<td>Action</td>
<td>Transforms data with a Create!form Transform project</td>
</tr>
<tr>
<td><img src="image" alt="Update Icon" /></td>
<td>Update</td>
<td>Action</td>
<td>Updates an external database</td>
</tr>
<tr>
<td><img src="image" alt="Get File Icon" /></td>
<td>Get File</td>
<td>Action</td>
<td>Terminates data flow and opens a new file</td>
</tr>
</tbody>
</table>
Table 6.1:

<table>
<thead>
<tr>
<th>Icon</th>
<th>Node</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Delete File Icon]</td>
<td>Delete File</td>
<td>Action</td>
<td>Deletes any file</td>
</tr>
<tr>
<td>![Run Program Icon]</td>
<td>Run Program</td>
<td>Action</td>
<td>Runs an external program</td>
</tr>
</tbody>
</table>

Note

• *A merge action can also be performed at a print queue (See Send node).*
• *A split action can also be performed at a Sort node.*

About Parent, Sibling and Child Relationships

Relationships between nodes are described respective to the selected or reference node, and can be either a parent, a sibling or a child. The types of node relationships are demonstrated in the diagram below:
When you insert a node in the Design Window, you can choose the type of relationship the new node has with the reference node by:

- Using the Insert menu
- By cursor positioning

For more information, refer to Insert Nodes.
Sequence and Timing

The sequence of processing is determined by the position of the node in the design. Processing starts at the root node at the top left corner and proceeds down and across each branch. Processing will not start in a lower branch until processing at the last (right-most) node in the branch above has ended. Processing at destination nodes ends as soon as the input file has been delivered to the folder, queue or URL.

When considering sequence and timing issues at a destination node, the time the job waits on the queue, and the time the job takes to be merged and output by the queue, does not affect the time at which processing starts at the next node.

Using Variables and Expressions to Control Actions

An expression combines and manipulates data from variables, which can be used to control actions or supply values to other processes. When the value returned by an expression is either “true” or “false”, the expression is referred to as a condition. You can use expressions and conditions to control how the input file is processed, including:

- To split the input file into subdocuments
- To filter what is processed at a node
- To change the destination of the output when certain conditions are met
- To select an action based on the properties of the input file
- To provide values for database updates
- To select which database should be updated
- To select another spool file
- To set options when running external programs

The types of variables available, and when to use them, are summarized below:
Use | To Provide
--- | ---
section variables | information derived from the input file (Data Variables, Derived Variables and Section Variables for more information)
user variables | user defined expressions that combine and manipulate other variables and values (refer to User Variables for more information)
global user variables | information that relates to the input file as a whole (refer to Global User Variables for more information)
system variables | page numbers, date, time, file and user attributes (refer to System Variables for more information)
lookup variables | information extracted from external databases (refer to Lookup Variables for more information)
environment variables | information derived from the production environment (refer to Environment Variables for more information)

For information on how to build expressions and conditions using these variables, refer to Chapter Nine, *Expressions and Conditions.*

For examples of how expressions can be used to control actions, refer to:

- Design Methods
- Filtering
- Splitting
- Sorting
- Duplicating
- Merging
- Transforming
- Defining Destinations
- Working with Job Tickets
- Using Multiple Create!form Director Projects
- Publishing
- Working with Databases
Adding Nodes

You can add nodes using the tools on the Design toolbar. The way in which the node will be added or inserted into the design will be indicated by the shape of the cursor. (Refer to Insert Nodes for more information.) There are some restrictions on where you can add certain types of nodes: the cursor indicates that you cannot add the selected node type at the current cursor position.

To create a node

1. From the Design toolbar, select the tool for the type of node you want to create.

2. In the Design Window, position the cursor over an existing node or connector. The shape of the cursor will change, indicating whether the new node will be inserted as a parent, sibling or child node. For more information on what these cursor shapes represent, refer to Insert Nodes.

3. Click to insert the node.

Table 6.2: Design Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Select</td>
<td>Select a node</td>
<td></td>
</tr>
<tr>
<td>Send</td>
<td>Send data to an output destination</td>
<td></td>
</tr>
<tr>
<td>Condition</td>
<td>Redirect by applying a condition</td>
<td></td>
</tr>
<tr>
<td>Split</td>
<td>Partition the input file as subdocuments</td>
<td></td>
</tr>
<tr>
<td>Sort</td>
<td>Sort input file partitions</td>
<td></td>
</tr>
</tbody>
</table>
Table 6.2: Design Tools

<table>
<thead>
<tr>
<th>Tool</th>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sub Project</td>
<td>Process data through another Create!form Director project</td>
<td></td>
</tr>
<tr>
<td>Merge</td>
<td>Merge data with a Create!form Designer project</td>
<td></td>
</tr>
<tr>
<td>Transform</td>
<td>Transform data with a Create!form Transform project</td>
<td></td>
</tr>
<tr>
<td>Update</td>
<td>Update an external database</td>
<td></td>
</tr>
<tr>
<td>Get File</td>
<td>Terminate data flow and open a new file</td>
<td></td>
</tr>
<tr>
<td>Delete File</td>
<td>Delete any file</td>
<td></td>
</tr>
<tr>
<td>Run Program</td>
<td>Run an external program</td>
<td></td>
</tr>
</tbody>
</table>
Selecting and Editing Nodes

You can edit your project in the Design Window by selecting, inserting, moving, copying and deleting nodes with the cursor.

What do you want to do?
- Select Nodes
- Insert Nodes
- Edit Nodes
- Move Nodes
- Copy and Paste Nodes
- Delete Nodes

Select Nodes

Use the Selection tool to select a node. Only one node can be selected at a time.

Insert Nodes

When you insert new nodes with the Design tools, you can control where the new node is inserted in one of two ways:

- by selecting a command from the Insert menu; the new node will be positioned relative to the selected node.
- by positioning the cursor over existing nodes and connectors; the new node will be positioned according to the current cursor shape; the selected node is ignored.

Using the Insert menu

1. Select the node you want to insert adjacent to.
2. Select a tool from the Design toolbar.
3. To insert the node, select a command from the Insert menu:
By cursor positioning

1. Select a tool from the Design toolbar.
2. Move the cursor over an existing node or connector until the required action is indicated by the cursor shape in the table below.
3. Click to insert the node.

Table 6.3:

<table>
<thead>
<tr>
<th>Cursor</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1" alt="cursor_shape" /></td>
<td>Insert as sibling before next node.</td>
</tr>
<tr>
<td><img src="image2" alt="cursor_shape" /></td>
<td>Insert as sibling after next node.</td>
</tr>
<tr>
<td><img src="image3" alt="cursor_shape" /></td>
<td>Insert as parent of next node.</td>
</tr>
<tr>
<td><img src="image4" alt="cursor_shape" /></td>
<td>Insert as &quot;Yes&quot; child of nearest node.</td>
</tr>
<tr>
<td><img src="image5" alt="cursor_shape" /></td>
<td>Insert as &quot;Yes&quot; child of previous condition node.</td>
</tr>
</tbody>
</table>
Table 6.3:

<table>
<thead>
<tr>
<th>Cursor</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Insert as &quot;No&quot; child of previous condition node.</td>
</tr>
<tr>
<td><img src="image" alt="Icon" /></td>
<td>Not allowed at cursor position.</td>
</tr>
</tbody>
</table>

Refer to about About Parent, Sibling and Child Relationships for more information.

**Edit Nodes**

The properties of the node can be displayed in several ways:

- By double-clicking the node.
- By right-clicking the node, and choosing Properties from the short-cut menu.
- By selecting the node, and choosing Properties from the Node menu.

Choose the method that best suits the way you work.

For more information about editing the properties of nodes, refer to Node Properties.
Move Nodes

Use the Selection tool to click-and-drag the node over the connector where you want to insert the node. Observe how the cursor changes shape to indicate how the node will be inserted, depending on which part of the connector the cursor is over. For more information on what these cursor shapes represent, refer to Insert Nodes.

Copy and Paste Nodes

You can use the following commands from the Edit menu, or shortcut menu to cut, copy and paste nodes from one location to another in the project.

<table>
<thead>
<tr>
<th>Tool</th>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Cut Icon]</td>
<td>Cut</td>
<td>Copy selected node to clipboard and delete.</td>
</tr>
<tr>
<td>![Copy Icon]</td>
<td>Copy</td>
<td>Copy selected node to clipboard.</td>
</tr>
<tr>
<td>![Paste As &gt; Sibling Before Icon]</td>
<td>Paste As &gt; Sibling Before</td>
<td>Paste before as sibling of current node.</td>
</tr>
<tr>
<td>![Paste As &gt; Sibling After Icon]</td>
<td>Paste As &gt; Sibling After</td>
<td>Paste after as sibling of current node.</td>
</tr>
<tr>
<td>![Paste As &gt; Parent Icon]</td>
<td>Paste As &gt; Parent</td>
<td>Paste before as parent of current node.</td>
</tr>
<tr>
<td>![Paste As &gt; Child Icon]</td>
<td>Paste As &gt; Child</td>
<td>Paste after as child of current node.</td>
</tr>
</tbody>
</table>
Selecting and Editing Nodes

To copy and paste a node

1. Select the node to copy.
2. To copy the node without deleting,
   - From the Main menu, select Edit > Copy, or
   - From the shortcut menu, select Copy, or
   - Press Ctrl+C
3. To copy and delete the node,
   - From the Main menu, select Edit > Cut, or
   - From the shortcut menu, select Cut, or
   - Press Ctrl+X.
4. Right-click the node where you want the copied node to be inserted before or after.
5. From the shortcut menu, select a command from the Paste As menu (see table above).

To drag and drop a copy of a node

1. While pressing the Ctrl key, click-and-drag the node away from the original.
2. Drag the node over the connector where you want to insert the copied node. Observe how the cursor changes shape to indicate how the node will be inserted, depending on which part of the connector the cursor is over. For information on what these cursor shapes represent, refer to Insert Nodes.

Table 6.4:

<table>
<thead>
<tr>
<th>Tool</th>
<th>Command</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Paste As &gt; Child if &quot;Yes&quot;</td>
<td>Paste as &quot;Yes&quot; child of condition node.</td>
</tr>
<tr>
<td></td>
<td>Paste As&gt; Child if &quot;No&quot;</td>
<td>Paste as &quot;No&quot; child of condition node.</td>
</tr>
</tbody>
</table>

To create a copy of a node
Delete Nodes

Use the Selection tool to select the node, then press the Del key, or select the Delete command from either the shortcut menu, or the Edit menu. All nodes downstream from the deleted node will also be deleted.
Node Properties

The properties of a node control the actions performed by the node and the location of destinations.

What do you want to know about?

- Send Node
- Condition Node
- Split Node
- Sort Node
- Sub Project Node
- Merge Node
- Transform Node
- Update Node
- Get File Node
- Delete File Node
- Run Program Node

For more information of how to apply these nodes in a project, refer to Chapter Seven, Methods and Procedures.

Send Node

Use the Send node to output the spool file to a destination. The destination can be a folder, URL, printer port (current device only), package and/or a print queue. To edit the properties of an existing node, double-click the node in the Design Window; the Send dialog will be displayed from which you can view and edit the properties of the node:
Figure 6.3: Send dialog with Folder as destination type
Figure 6.4: Send dialog with Package as destination type

**Description**
The node name, supplied by the user.

**Destination Type**
The destination can be either the current device, a network folder, an FTP address, an HTTP address, package and/or a print queue. For more information, refer to Types of Destinations.

**Image**
Click the Browse button to display the Select Image dialog, from which you can select a different icon and import new images.

**Package Destination Type**
This will save the current data stream to memory and append to any other previously saved package data for future sending to a physical destination type. The package will get reset every time a new set of data is loaded into Director or when Empty Package after Send is selected. This is typically used
when using Merge Nodes to create each page of a document because it will allow the concatenating of all output (Postscript) into one print stream or the conversion to one output document.

**Select Named Destination**

- To specify an explicit queue name, pathname or address for the destination, clear the checkbox.
- To select the queue name, pathname or address from a list of pre-defined named destinations, select the checkbox.

**Folder Name**

*(Destination type is Folder and Select named destination option is cleared)*

The pathname of the destination folder is displayed. This may be defined by an expression, or by the actual pathname. For information on defining folder names, refer to Defining Destinations.

**Destination Name**

*(Select named destination option is selected)*

The name of the destination is displayed. From this field you can:

- Select New to define a new named destination
- Select Manage to manage the list of named destinations
- Select a different named destination from the list
- Edit the current named destination by clicking the Edit button

Refer to Defining Named Destinations for more information.

**Document Name**

The name of the destination file is displayed. This may be defined by an expression, or by the actual file name. To create or edit an expression, click the Expression Builder button. Refer to Expressions and Conditions for more information.

Refer to Naming Destination Documents for more information.

**Append to File**

*(Destination type is Folder)*

- To overwrite an existing file, clear the checkbox.
- To append to an existing file, select the checkbox.
Send from Package
This option will take the current datastream that is in the memory package and send that to whatever physical destination is chosen.

Output Format
The choices are Postscript, which is the default, PDF or PCL. This will take the Postscript that is in the package and convert it to PDF or PCL. Note that all package data should be postscript or text.

Empty Package after Send
This option will clean out the package cache, which will also happen when each new set arrives in Director.

Copies
• **Destination type** is Queue
  The number of copies generated at the destination queue is displayed. This may be defined by an expression, or by typing in the number of copies. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.
• **Destination type** is Package
  Copies the current data stream (output page(s)) the specified number of times.

Job Ticket Only
Select the checkbox to output only the information contained in the job ticket using the template specified in the Template field. Refer to Working with Job Tickets for more information.

Template
The template used to create the job ticket is displayed. From this field you can:
• select (none) to output without a job ticket
• select New to create a new job ticket template
• select an existing job ticket template from the list
• edit the current template by clicking the button
Refer to Working with Job Tickets for more information.
Note

The default template Default.jtt will prevent Recursion of the input file.

Parameter

The job ticket parameters defined in the current template are displayed. To add or remove parameters from the template, click the button next to the Template field. Refer to Working with Job Tickets for more information.

Expression

The expressions that define the values of job ticket parameters in the current template are displayed. To edit the expression, click the Expression Builder button adjacent to the expression. Refer to Building Expressions for more information.

User Variables

Click the User Variables button to edit and create user variables used at this node.

Preview Node

This menu item is available when you right-click on a Send Node. When selected and conditions allow, it will create a preview PDF for the chosen Send Node.
Note

*Depending on the input file and the conditions for the selected Director project, this option might not return any preview.*

## Condition Node

The input file is directed to one of two possible processing paths depending on whether the specified condition is true or false. If true, the file is sent to the “Yes” child node, if false, to the “No” child node. Refer to About Parent, Sibling and Child Relationships for more information on node relationships.

### Description

The node name, supplied by the user.

### Condition

The logical expression that determines which processing path the file is sent to. To create or edit the expression, click the Condition Builder button.

Refer to Building Conditions for more information.

### User Variables

Click the User Variables button to edit and create user variables used at this node.

## Split Node

The input file is partitioned into sub-documents. Each partition is processed sequentially downstream from the Split node and may be filtered into different processing paths at Condition nodes.

### Description

The node name, supplied by the user.

### Split by

Select the method by which the file will be partitioned:

- Project set
The file will be split into sets, which is defined by the Project>Define Set command. Refer to Chapter Five, Input Design, for more information about defining sets for different types of input files.

- **Condition**
  The file will be split when the condition tests true. For example, you can test the value of a data variable or a user variable as it changes section by section or page by page. To create or edit the expression, select the radio button and click the Condition Builder button. Refer to Building Conditions for more information.

**Sort Node**

The input file is sorted by partition using one or more sorting levels.

**Description**

The node name, supplied by the user.

**Partition By**

Select the method by which the file will be sorted:

- **Project set**
  The file will be sorted by set, which is defined by the Project>Define Set command. For more information about defining sets for different types of input files, refer to Chapter Five, Input Design.

- **Condition**
  The file will be sorted into partitions determined when the condition tests true. For example, you can test the value of a data variable or a user variable as it changes section by section or page by page. To create or edit the expression, select the radio button and click the Condition Builder button. Refer to Building Conditions for more information.

*(sorting options)*

Select the method by which the file will be sorted:

- **Level**
  The file can be sorted into one or more levels. For example, you can set Level 1 to sort sets into States, and Level 2 to sort the sets within each State by their Zip codes.
Node Properties

- Sort on variable
  The data variable you want to sort or group by.
- Sort order
  Select either Ascending or Descending order.
- Sort mode
  The following table demonstrates the ASCII, regional and numeric sort modes.

Table 6.5:

<table>
<thead>
<tr>
<th>Unssorted Sample</th>
<th>Sort Mode</th>
<th>ASCII</th>
<th>Regional</th>
<th>Numeric</th>
</tr>
</thead>
<tbody>
<tr>
<td>(space)</td>
<td>(space)</td>
<td>(space)</td>
<td>(space)</td>
<td></td>
</tr>
<tr>
<td>02 02 02 A</td>
<td>1 0 11B</td>
<td>1 1 01 0C</td>
<td>222a</td>
<td></td>
</tr>
<tr>
<td>333c</td>
<td>AAA1</td>
<td>BBa 0 2</td>
<td>CCB2</td>
<td></td>
</tr>
<tr>
<td>3 0 2</td>
<td>aaC</td>
<td>CCC1 0</td>
<td>cccc1 0</td>
<td></td>
</tr>
<tr>
<td>A A</td>
<td>A A</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>B B</td>
<td>a a</td>
<td>02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>C C</td>
<td>B B</td>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>a a</td>
<td>C C</td>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>c c</td>
<td>c c</td>
<td>10</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

The sort order when using regional mode can vary with your Windows locale.

Split by Partition

You can optionally split the sorted file into the same partitions used for sorting. Refer to Split Node for more information.

Sub Project Node

The input file is passed to another Create!form Director project for processing and then passed back to the current project.

Description

The node name, supplied by the user.
**Director project**

The name of the Create!form Director project is displayed. This may be defined by an expression, or by the selecting the project file with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

To open the selected project in Create!form Director, click the button.

**User Variables**

Click the User Variables button to edit and create user variables used at this node.

---

**Merge Node**

An in-line merge is performed allowing the merged output to be processed further downstream. This performs the same function as a Send to (merge) Queue node, except the Send node terminates processing of the file on that branch.

**Description**

The node name, supplied by the user.

**Form Project**

The name of the Create!form Designer project is displayed. This may be defined by an expression, or by the selecting the project file with the browse button. Refer to Building Expressions for more information.

To open the selected project in Create!form Designer, click the button.

If the selected form project is a JDE 3.X Project, check the Choose JDE PDF 3.X Project option below the Form project dropdown list.

**Configuration**

The merge configuration used in the merge is displayed. Refer to Merge Configurations for more information.

**Additional Merge Parameters**

The job ticket attached to the merged output is defined in this section. These Job Tickets can also be used for passing parameters to the merge engine and they can be referenced via the Create!form Designer project. An example of why you might want to do this would be for page numbering when merge nodes are used with different projects for each page.
Note

*These parameters are only used in the creation and output of PDF documents to define security options and number of copies.*

---

**User Variables**

Click the User Variables button to edit and create user variables used at this node.

**Transform Node**

An in-line transform is performed allowing the transformed output to be processed further downstream. This performs a similar function to the Merge node, except the input file is transformed into a new format e.g. CSV or XML, instead of being merged with a form.

**Description**

The node name, supplied by the user.

**Transform project**

The name of the Create!form Transform project is displayed. This may be defined by an expression, or by the selecting the project file with the browse button. To create or edit an expression, click the Expression Builder button. Refer to [Building Expressions](#) for more information.

To open the selected project in Create!form Transform, click the button.

**User Variables**

Click the User Variables button to edit and create user variables used at this node.

**Update Node**

Adds and updates records in a database. The Update node is typically used in conjunction with the Transform node (to create a CSV update file) and the Get File node (to open the update file).

Before you can select the database from the Update Database dialog, you must first create a DSN.
**Description**
The node name, supplied by the user.

**Database**
The DSN details are displayed.

**Connection String**
The connection string is generated for you when you select the DSN. To select the DSN, click the Select DSN button adjacent to the Connection string field and supply your login details when prompted.

**Table**
Select the table you want to update.

**Action**
The action performed by the update is displayed. Select from the options displayed:

- **Add only**
  Use this option to add new records to the database without changing any existing database records.

- **Update only**
  Use this option to update existing records in the database without adding any new database records.

- **Add/Update**
  Use this option to update existing records in the database and add any new records.

**User Variables**
Click the User Variables button to edit and create user variables used at this node.

**Get File Node**
The current input file will be terminated and a new input file is opened and processed on the current branch.
Description
The node name, supplied by the user.

Folder Name
The pathname of the target folder is displayed. This may be defined by an expression, or by the selecting the folder with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

File Name
The name of the target file is displayed. This may be defined by an expression, or by the selecting the file with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

Keep Trying For... Seconds
The action will time-out if it cannot be performed within the specified time period.

User Variables
Click the User Variables button to edit and create user variables used at this node.

Delete File Node
The specified file will be deleted.

Description
The node name, supplied by the user.

Folder Name
The pathname of the target folder is displayed. This may be defined by an expression, or by the selecting the folder with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

File Name
The name of the file to be deleted is displayed. This may be defined by an expression, or by the selecting the file with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.
**Keep Trying For...Seconds**

The action will time-out if it cannot be performed within the specified time period.

**User Variables**

Click the User Variables button to edit and create user variables used at this node.

---

**Run Program Node**

An external program is run using the options defined in the command line.

---

**Note**

Processing at the next node in the project will occur immediately, without waiting for a response from the external program.

---

**Description**

The node name, supplied by the user.

**Executable File**

The name of the program file is displayed. This may be defined by an expression, or by selecting the file with the browse button. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

**Command Line**

The command line options used by the program are displayed. This may be defined by typing the actual text string, or by an expression. To create or edit an expression, click the Expression Builder button. Refer to Building Expressions for more information.

**Keep Trying For... Seconds**

The action will time-out if it cannot be performed within the specified time period.

**User Variables**

Click the User Variables button to edit and create user variables used at this node.
Window Display Preferences

You can customize the way nodes are edited and displayed by selecting the Tools>Preferences command. With the options displayed in the panes of the Preferences dialog, you can:

• Turn on and off rulers
• Turn on and off tooltips
• Select how nodes are annotated
• Customize the color of connectors, annotation and the window background
• Customize node icons
Chapter 7: Methods and Procedures

The design process involves combining the different types of nodes available to achieve a desired workflow. Create!form Director allows you to customize your design to cope with different input formats, processing methods and destination types.

The following topics are covered in this chapter:

- Input Options
- Key Tasks
- Output Options
Input Options

The input file type, the structure of the input file and how its contents are mapped as variables are defined by the project DataMap. You can create a DataMap for each project, or share DataMaps with other Create!form projects, including Create!form Designer and Create!form Transform projects. Refer to DataMap Types and Options for more information.

The DataMap of a Create!form Director project is specified independently of any merge projects or Create!form Director sub-projects that may be used in the project.

To create a new DataMap for a project, refer to the following topics:

- Designing for CSV Input Files
- Designing for Text Input Files
- Designing for XML Input Files
- Designing for JDE PDF Input Files
Key Tasks

What do you want to know about?

- Design Methods
- Filtering
- Splitting
- Sorting
- Duplicating
- Merging
- Transforming
- Defining Destinations
- Working with Job Tickets
- Using Multiple Create!form Director Projects
- Publishing
- Working with Databases
- Triggering

Design Methods

Each node in the project design defines a specific task that you want to perform on the input file. For example, you may wish to split the input file into separate invoices, so that each split can be merged with a particular invoice form, and delivered to a particular printer. You can design your project so that the same task is performed on every input file, and delivered to the same destinations, or you can define conditions and expressions so that the project treats input files in different ways, depending on the contents and properties of the file.

By way of example, consider the above case where the input file needs to be split by invoice, merged with different form projects and sent to multiple destinations. This can be achieved in a variety of ways, depending on the structure of your input file. In this particular case, assume that a DataMap has been created, and the DataMap defines individual invoices as sets, and the zip code on the first page of each invoice has been mapped as a data variable
named Zip_Code. In this case, we also know that the input file always arrives with a job ticket that has been attached by Create!send, which provides information supplied by the originating application. The parameters in this particular job ticket are the host system name (%cpSystem:), the user name (%cpUser:), the input file name (%cpFileName:) and the project name (%cpParam:-s):

Figure 7.1: Sample job ticket

Here are three different methods you can use to create your design, which achieve exactly the same outcome:

**Use condition nodes with multiple send nodes**

1. Split the input file by invoice. (See Splitting for more information.)
2. Use condition nodes to direct invoices to different send nodes. (See Filtering for more information.)
3. Merge each split with the required region invoice form. (See Merging for more information.)
4. Output to the required printer in each district office. (See Defining Destinations for more information.)
In this case, the Condition nodes use expressions that test the value of Zip_Code in each invoice partition, which are then directed to the required Send node. The merge project and the destination are explicitly defined at each Send node.

**Use an expression to select merge projects and use condition nodes to select destinations**

1. Split the input file by invoice. (Refer to Splitting for more information.)
2. Send all sets to a merge node and use an expression to select the required merge project. (Refer to Working with Job Tickets for more information.)
3. Use condition nodes to direct merged invoices to different Send nodes. (Refer to Filtering for more information.)
In this case, the \%cpUser: job ticket variable is used in an expression that defines the name of the merge project (\%cpUser:Western means that the spool file only contains “Western” region invoices), and the merged output is filtered on Zip_Code through Condition nodes to the required Send node. The destination is explicitly defined at each Send node.

**Use expressions to merge and send from the same node**

1. Split the input file by invoice. (Refer to Splitting for more information.)

2. Send all invoices to a send node with one merge project that employs variable subforms and alternate sections to customize the form for each regional office. For more information, refer to Create!form Designer user manual.

3. Use an expression to select a destination (Refer to Defining Destinations).
In this case, the %cpUser: job ticket variable is used by the merge project to select the required letterhead subform for each regional office, and in the expression that constructs the name of the destination print queue.

**Filtering**

Condition nodes act as filters on the spool file so that you can direct either the entire spool file, or parts of the spool file to different nodes downstream for further processing. Consider the two projects shown in the diagram below:

In project A, the test will be applied to the entire spool file, which will be directed to one of two child nodes. Typically the test would be applied on the value of a global variable or a job ticket variable, which identifies how the spool file should be processed. For example, the job ticket variable %cpOutQName might have two possible values “Invoices” or “Dockets”. In
this case, you can use a Condition node with the condition \%cpOutQName=="Invoices" which, if true, will direct the spool file to the Invoices sub-project, and, if false, to the Dockets sub-project.

In project B, the spool file is first split into sets, and the test will be applied to each set in the spool file. Typically the test would be applied on the value of a data variable or user variable defined in the DataMap, which changes with each set in the spool file. For example, a data variable Region_Code might be defined in the DataMap, which has five possible values “USA”, “CAN”, “CH”, “UK” or “AUS”. In this case, you can use a Condition node with the condition Region_Code=="USA" OR Region_Code=="CAN" which, if true, will direct the spool file to the NAM merge queue, and, if false, to the INT merge queue.

The variables available to you when creating conditions are listed in the Variable section of the Condition Builder, which include:

- Section variables: data variables and derived variables defined by the DataMap.
- User variables: user defined expressions that can vary by node or set.
- Global user variables: user defined expressions that relate to the entire the spool file e.g. global variables may be constant or can accumulate values set to set, or node to node.
- Lookup variables: values extracted from external databases.
- System variables: system defined values relating to the spool file, sets, the server and other information.
- Job ticket variables: values derived from the spool file job ticket that apply to the entire spool file.
- Environment variables: user defined values that apply to the entire production system, or to specific destinations.

To create a condition

- Click the Condition Builder button on the Condition dialog. Refer to Chapter Nine, Expressions and Conditions, for more information.

To create a user variable you can use in a condition

- Click the User Variables button on the Condition dialog.

The User Variables dialog will be displayed. Refer to Chapter Nine, Expressions and Conditions, for more information.

Refer to Chapter Eight, Variables, for more information.
Splitting

A common reason for using a split action is to distribute different parts of a document to different destinations. For example, you may want to send the data relating to particular states to different print queues. When a spool file is split, the variables associated with the spool file, like job ticket variables and global variables, will remain available to each part of the file as they are processed downstream from the split node.

You can split the spool file by set, or by defining a condition:

**Splitting by Set**

Sets are defined by the project DataMap. Refer to Chapter Five, Input Design, for more information.

**Splitting by Condition**

When you use a condition to split the spool file, the split occurs when the condition tests true. For example, you might test when the page number equals 1 with the expression: `Page_No == "1"` or when the invoice number changes with the expression: `changed("Inv_No")`.

In these examples, the section variables `Page_No` and `Inv_No` would be defined in the project DataMap. The variables available to you when creating conditions are listed in the Variable section of the Condition Builder.

Splitting can also be used to group sets of documents together. For example, if you are outsourcing your printing, you might need to limit the size of each print batch. The split needs to occur at the start of the first set that follows a specified page limit. You can create a global variable in the merge project that is reset to 1 when both of these conditions are met, and then test when this occurs at the split node.

**About the Order in which Splits are Processed**

When the spool file is split, each part of the file will be processed in turn: the project will start processing the next part as soon as the preceding part has been delivered to a Send node. If the parts of the file are delivered to different print queues, the order in which printing each part is completed will depend on the number of print jobs in each queue, the size of each print job, and the relative speed of each printer. Refer to Sequence and Timing for more information.
Note

*You can also split the spool file at a Sort node. Refer to Sorting for more information.*

Sorting

You can re-order the contents of the spool file using a sort node. Typically, this is done so that sets are presented to the printer in a particular order, or to group sets together before the spool file is split. When sorting a spool file, you must specify how the file will be partitioned, and how these partitions will be ordered. For example, you can partition the file by sets, and then order the sets using a section variable, like an invoice number or customer code. Alternatively, you can perform a multi-level sort that orders the sets, first by zip code, and then alphabetically by customer name. The variables you use to control the sorting must be defined in the project DataMap.

Sort and Split

You can also split the spool file at a sort node if the sorting and splitting partitions are the same. For example, you can partition the spool file by set, sort by a date variable, and then split by set. To sort and split with different partitions, sort the spool file at a sort node, then split the file at a split node:
Duplicating

Duplication is commonly used when the same spool file needs to be output in multiple formats to multiple destinations. For example, when a document is printed to a PostScript printer, you can also create a PDF copy of the document for archiving.

You can duplicate the spool file by inserting multiple sibling nodes. Consider the example in the diagram below:

![Diagram](image)

In this project, the spool file is processed at node 1 (the parent node), then at node 2 (a child of node 1) and then at node 3 (a child of node 2). When node 4 is inserted as a sibling of node 2, the spool file will be duplicated and also processed at node 4. The duplicate is processed at node 4 as soon as processing has completed on all child nodes of higher sibling nodes i.e. when processing at node 3 is completed.

---

**Note**

To duplicate pages at the printer, modify your merge project using the Project>Properties command in Create!form Designer. For more information, refer to the Create!form Designer user guide.
Merging

Merging is the process by which a spool file is reformatted and reorganized with a Create!form Designer project. For information on how to create merge projects, refer to the Create!form Designer user guide.

A Create!form Designer project allows you to create customized forms in which the contents of the input file will be reformatted and reorganized, with additional text, graphics, barcodes and other objects. Typically, Create!form Designer projects are used for invoices, picking slips, letters, delivery dockets and other forms requiring a graphical layout.

DataMaps

When you design a Create!form Designer project, you must also define a DataMap, which describes the structure and contents of the input file, so it can be mapped to the output form. The DataMap you create or select for your Create!form Designer projects is independent of the DataMap you create or select for the Create!form Director project, but they can be shared. Refer to DataMap Types and Options for more information.

In-line Merge

You can use a Merge node to merge the input file with a Create!form Designer project and then continue processing the merged output in the Create!form Director project.

Merging at a Queue

You can use a Send node to merge the input file with a Create!form Designer project at a Create!form Server merge queue.

Merge Configurations

Merge configurations are named configuration settings that define the merge and output settings used by Create!form Server. You can create merge configurations in any Create!form project i.e. Create!form Director, Create!form Designer and Create!form Transform projects. Project merge configurations will be copied to the server when the project is transferred.
Transforming

A Create!form Transform project allows you create text output files in which the contents of the spool file is reorganized and reformatted. Typically, Create!form Transform projects are used for creating CSV files for database updates, XML transaction files and other formatted text messages. For example, with Create!form Transform you can:

- Extract summary data from the spool file, and output the data as a CSV file to perform a database update.
- Generate usernames and passwords, which are sent to customers by email to gain access to protected PDF statements.

For information on how to create Transform projects, refer to the Create!form Transform user guide.

Defining Destinations

The destination of the processed and merged spool file is defined at the Send node. The way in which you define destinations will depend on the type of destination you are sending the spool file to.

Once you have selected the destination type, you may be required to provide additional information for:

- Naming Destination Documents
- Defining Login Details

You can then define the destination in several ways:

- Defining Destinations by Address
- Defining Named Destinations
- Defining Destinations by Expression
Types of Destinations

The spool file can be sent to the following destination types:

- **Current device**
  A Create!form Director project is called from a specific print queue. When the print queue is created, a printer port is defined, such as a network printer or a Create!form fax, email or archive port. The current device of a Create!form Director project is the printer port of the print queue from which the project is called.

- **Folder**
  A local or network folder, where the spool file can be saved to file.

- **FTP**
  An FTP address, where the spool file can be saved to file.

- **HTTP**
  An HTTP address, where the spool file can be saved to file.

- **Queue**
  A local or network print queue, where the spool file can be
  - Merged with a Create!form Designer or Create!form Transform project and printed to a print device or a Create!form port, or
  - Processed with another Create!form Director project, or
  - Printed without merging or further processing.

- **Package**
  Saves the current data stream to memory and appends to any other previously saved package data for future sending to a physical destination type. Refer to Send Node for more information.
**METHODS AND PROCEDURES**

**Key Tasks**

**Naming Destination Documents**

Certain types of destinations allow you to name the file that will be created at the selected destination folder or address.

**To define an explicit file name**

Type the full pathname in the file name field.

**To define the file name by expression**

You can use an expression to define the destination file name. Click the Expression Builder button. The expression must generate the full pathname, and account for exceptions that might occur in the input data. Refer to Building Expressions for more information.

**If you do not specify a document name**

If a document name is required, and has not been specified, Create!form Director will automatically generate file names in the form stem-nnnnn.xxx, where stem is the stem of the input file name, nnnnn is a sequential file partition number, and xxx is the file name extension determined by the output format, e.g. StdInvoice-00234.pdf.

---

**Note**

*If the named file already exists at the destination, select or clear the Append to existing file checkbox to append the data to the existing file, or overwrite it.*

---

**Defining Login Details**

Certain types of destinations require you to specify a login name and password to view or write to the selected destination folder or address. When prompted, type the Login name and Password in the Login dialog.

**Defining Destinations by Address**

The options displayed on the Send dialog will vary depending on the type of destination you have selected.
**To select a printer**
1. Double-click the Send node. The Send dialog will be displayed.
2. From the Destination type dropdown list, select Queue.
3. Clear the Select named destination checkbox.
4. Click the browse button next to the Queue name field. The Select Printer dialog will be displayed.
5. From the Name field, select the printer name and click OK.
6. Click OK.

**To select a folder**
1. Double-click the Send node. The Send dialog will be displayed.
2. From the Destination type dropdown list, select Folder.
3. Clear the Select named destination checkbox.
4. Click the browse button next to the Folder name field. The Select Destination Folder dialog will be displayed.
5. Select the folder, or create a new folder, and click OK.
6. If required, define the document name (Naming Destination Documents) in the Document name field.
7. Click OK.

**To select FTP and HTTP addresses**
1. Double-click the Send node. The Send dialog will be displayed.
2. From the Destination type dropdown list, select FTP or HTTP.
3. Clear the Select named destination checkbox.
4. In the FTP/HTTP address field, type the full address, e.g. ftp://www.newworld.com/private.
5. Select the folder, or create a new folder, and click OK.
6. If required, define the document name (Naming Destination Documents).
7. If required, define the login name and password required (Naming Destination Documents) to access this address.
8. Click OK.
Defining Named Destinations

Named destinations are provided for ease of identification of print queues, folder pathnames and URL addresses. Instead of selecting from a list of obscure printer addresses like “\Flyingfox\Lexmark Optra S 2450 PS”, you can give each destination a meaningful name like “HR Dept”, “Fax server” and “Archive folder”.

Named destinations only need to be created once in the project and will be available in all send nodes of the same type. Named destinations can be assigned to all destination types: folder, FTP, HTTP and queue.

Working with named destinations on the server

Named destinations are saved to the configuration file destinations.data in the \SystemResource folder. When the project is transferred to the server, the destinations file on the server will be overwritten. It is recommended practice, before you begin defining named destinations, to copy the destinations file from the server to the design workstation. This will ensure that, when the project is transferred, any new destinations will be added and all existing destinations will be preserved.

To create a named destination

1. In the Design Window, double-click the send node where the destination is will be specified. The Send dialog will be displayed.
2. Select the Select named destination checkbox.
3. From the Destination name drop-down list, select New...The <destination type> Destination dialog will be displayed.
4. Click the Browse button next to the <destination type> name field. The Select Printer dialog will be displayed.
5. From the Name dropdown list, select the printer e.g. \server.co.hr5\HP4050T, and click OK.
6. In the Destination name field, type the name you want to use for this destination e.g. Denver Accounts, and click OK.
7. Repeat steps 3 through 6 until each destination has been named.
**To use a named destination**

1. In the Design Window, double-click the send node where the destination is specified. The Send dialog will be displayed.
2. Ensure the Select named destination checkbox is selected.
3. From the Destination name dropdown list, select the destination name.

**Defining Destinations by Expression**

You can use an expression to select the destination. For example, say the job ticket variable `%cpParam:-s` has values of “Denver”, “Seattle” or “Phoenix”, which is used to select the merge project at a Send node e.g. Seattle.cfp6; and the output from each of these projects is sent to different print queues. In this case, you would create a destination variable whose value was conditional on the value of the `%cpParam:-s` job ticket variable.

The steps involved in this particular case would be

1. Double-click the send node where the merge occurs. The Send dialog will be displayed.
2. Click the User Variables button. The User Variables dialog will be displayed.
3. Define the user variable `Print_To`, whose value is conditional on the value of the `%cpParam:-s` job ticket variable, as shown in Figure 7.2.

![User Variables (Merge queue)](image)

*Figure 7.2: Defining a destination variable*
4  Click OK to close the User Variables dialog.
5  On the Send dialog, clear the Select named destination checkbox.
6  Click the Expression Builder button next to the Queue name field. The Expression Builder dialog will be displayed.
7  From the Variable field, select the user variable Print_To.
8  Click OK.

Working with Job Tickets

A job ticket can be added to the input file, and is used to pass information to other Create!form components when merging, faxing, emailing and archiving the output. The information required by the output component is defined by a job ticket template. Standard templates for printing, faxing and emailing are provided, but you can customize these templates for specific output tasks and types of data.

Default Template

When a new Send node is created, the template Default.jtt is initially selected. This template contains the %cpBegin and %cpEnd tags, which causes the %cpOrigProject tag to be added when processed by Create!form Server. This tag tells the server that the file has already been processed as a safeguard against recursion.

For information about the content and format of templates, refer to the user guides for the respective output components.

Using Multiple Create!form Director Projects

You can pass the input file to another Create!form Director project at a Sub Project node. This is typically used in conjunction with Condition nodes to determine which Create!form Director project the input file should be processed by:
Publishing

Publishing is generating output through an inline merge node to a file, HTTP, or FTP site. A publishing license is required when producing any of the outputs described above.

Note

An Unlimited printer license is needed is you are using an inline merge node and sending output to a queue.

Working with Databases

You can extract information from databases for use in the project, as well as update databases from the project.

Extracting information from a database for use in the project

You can use lookup variables at nodes in the project to extract information stored in an external database. Refer to Lookup Variables for more information.
**Updating a database with information extracted from the spool file**

You can use an Update node to add and update records in a database. The Update node is typically used in conjunction with the Transform node (to create a CSV update file) and the Get File node (to open the update file).

**Triggering**

You can trigger other workflow events in several ways. For example, you might send an email, run an external program, or send a file to a watch folder. These actions can be initiated by the Run Program and Get File nodes. Consider the examples below, in which different actions are triggered after a spool file merge has been completed:

In project A, once the spool file has been merged and sent to the print queue, a message file is opened by the Get File node and sent by email as a notification.
In project B, as soon as the send node prints a delivery docket, the materials safety data sheet is printed to the same printer by running MS Word in command line mode from a Run Program node.

When triggering actions, it is important that they are performed in the correct sequence. Refer to Sequence and Timing for more information.

By combining these basic mechanisms with other Create!form Director functions, you can customize your projects to trigger a wide range of workflow processes, actions and events, as demonstrated in the diagram below:
METHODS AND PROCEDURES

Key Tasks

1. Create!form Director
2. Publish customer statement to web site
3. Send
4. Notify customer by email
5. Get File
6. Transform
7. Update Database
8. Generate username and password
9. Send

New Customer?
Output Options

What do you want to know about?

- Output Formats
- Recursion
- Output to File
- Printing
- Fax
- Email
- Archiving

Output Formats

The output format from a Create!form Director project is determined by the processes defined in the project. A wide variety of formats can be generated, including text, PDF, CSV, PostScript or PCL, and depends on the format required at the destination device or output port.

The output format can be controlled at the following types of nodes:

- Merge node
  The output format is determined by the merge configuration specified at the node.
- Transform node
  The output format is determined by the Create!form Transform project specified at the node.
- Send to Queue node
  The output format is determined by the merge configuration specified at the queue.
Recursion

Recursion is the process by which a project can call itself. This can occur when the output is sent to the same queue from which the project was called. You can prevent recursion by selecting the default template (Default.jtt) at a Sub Project Node. When Create!form Server encounters the default template, the project setting at that queue will be ignored and the file will be output.

Output to File

Use the following destination types at a Send node to output to file:

- Folder
- FTP
- HTTP

Printing

Specify the output format in the merge configuration at Merge and Send nodes:

- PostScript
- PCL
- PDF

To print to the current device

1 Double-click the Send node in the Design Window. The Send dialog will be displayed.

2 In the Destination type field, and select Current Device from the drop-down.

To print to a different print device

1 Double-click the Send node in the Design Window. The Send dialog will be displayed.

2 In the Destination type field, and select Queue from the drop-down.
Output Options

**Fax**

Use a Send to Queue node to output to a Create!fax port. For more information, refer to the Create!fax user guide.

**Email**

Use a Send to Queue node to output to a Create!email port. For more information, refer to the Create!email user guide.

**Archiving**

You can send the output to Create!archive by either:

- Use a Send to Folder node to output to a Create!archive watch folder.
- Use a Send to Queue node to send the output to a Create!archive port.
CHAPTER 8

VARIABLES

You can use variables as arguments in expressions and conditions that control how data is processed in a project.

The following topics are covered in this chapter:

• About Variables
• Data Variables, Derived Variables and Section Variables
• User Variables
• Global User Variables
• System Variables
• Lookup Variables
• Environment Variables
• Job Ticket Variables
About Variables

Variables can be used as arguments in conditions and expressions defined in project nodes. There are several different types of variables that you can use in a project: the type of variable you use will depend on where you want to source your data and how it will be applied in the project:

- Data Variables, Derived Variables and Section Variables
- User Variables
- Global User Variables
- System Variables
- Lookup Variables
- Environment Variables
- Job Ticket Variables
Data Variables, Derived Variables and Section Variables

Data variables and derived variables are created in the Input Design and are available as Section Variables at nodes. You can view the variables defined in the input design from the Input Tree.
User Variables

User Variables are defined by users and derive their value from functions and expressions using other variables as arguments. User variables are created at and belong to a node. They can be used for a wide variety of tasks.

For example, you might create a user variable to construct an output filename. The user variable can take data from the input file, and use the text functions available to create a unique filename.

To create a user variable

1. Select the node you want the user variable to be available in.
2. From the Main menu, select Node > User Variables. The User Variables dialog will be displayed.
3. If required, click the Add Row button. A new variable is created.
4. To create a condition, click the Condition button in the Condition column. Refer to Chapter Nine, Conditions, for more information.
5. In the Variable Name column, type a name for the user variable.
6. Click the Expression button to create the expression that will generate the variable value. The Expression Builder will be displayed. Refer to Using the Expression Builder for more information.
7. Click OK.

Once you have created a user variable, you can use it to build other expressions or conditions defined from the node dialog.
Global User Variables

Global user variables enable you to create a variety of dynamic fields and use them in any node in the project. Global user variables have an initial value and are passed sequentially from node to node. At each node they can have their value changed.

Global user variables are typically used in conjunction with user variables. Global user variables can provide values for user variables, or derive values from user variables.

**To create a global user variable**

1. From the Main menu select Project > Global User Variables. The Global User Variables dialog will be displayed.

2. Click the Add Row button.

3. In the Variable Name column, type a name.

4. In the Initial Value column, type an initial value.

5. Click **OK**.

Global variables can be used in conditions and expressions at any node.
System Variables

Create!form Director provides pre-defined variables that you can use to determine information about the project and the operating environment. The following table describes the system variables available in the input and output of the design:

Sys.DocPageNum

<table>
<thead>
<tr>
<th>System Variable</th>
<th>Description</th>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys.DeviceName</td>
<td>Name of destination port, queue or device</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.DocName</td>
<td>Name of spool file</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.DocPageNum</td>
<td>Current page number in the document</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Sys.DocSetNum</td>
<td>Current set number in the document</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Sys.Hostname</td>
<td>Name of host server where queue resides</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.OSVersion</td>
<td>Version of operating system running on host</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.PreviousSection</td>
<td>Name of previous detail section</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Sys.ProductVersion</td>
<td>Details about Create!form Server installed on host</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.ProjectName</td>
<td>Name of current project</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.QueueName</td>
<td>Name of print queue where merge occurs</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.RandomString</td>
<td>Returns a random test string</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.SplitPageNum</td>
<td>Current page number in the split</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>Sys.SpoolID</td>
<td>Spool file ID on the merge queue</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.Time</td>
<td>Time of merge supplied by the system</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.Today</td>
<td>Current date value supplied by the system</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Sys.Username</td>
<td>Username property of spool file</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

System variables can be used in expressions and conditions at any node.
Lookup Variables

Lookup variables retrieve values from external databases. For example, you can use lookup variables to retrieve a customer telephone number from a database. Lookup variables are defined at and belong to a node in the project.

Requirements

To use lookup variables, you require a DSN connection to an ODBC database. To enable production printing, you must also create a permanent connection between the e-forms server and the database(s). For more information, see the Create!form Server User Guide.

To create a lookup variable

1. From the Create!form Director main menu, select Node>Lookup Variables. The Lookup Variables dialog will be displayed.

![Lookup Variables dialog](image)

*Figure 8.1: The Lookup Variables dialog*
2 From the drop-down list under the Database (DSN) field, select the DSN for the database you want to retrieve data from, or click New to create a new DSN. Setup procedures for databases will vary.

Notes

• When using the lookup feature to retrieve information from a file, the DSN name needs to contain the word file in lower case letters.

• Microsoft supplies an ODBC driver, however, we recommend using Data Direct drivers. Order them online at www.datadirect.com.

3 If multiple tables/views exist in the database, select the appropriate one from the Table/view drop-down.

4 In the Select records where field, define a query by selecting:

• The Database Field from the database table that will be searched in and;

• The Variable from the project containing the value that will be searched with. Ensure the query will only match with one record. If you need to add further qualifiers, clicking the Add Row button and select further fields and variables.

5 In the Assign to variables field, define the response to the above query by:

• typing the Variable Name in the project to assign to.

• selecting the Database Field from the database table to assign from.
Environment Variables

Environment variables are common to all projects in the production environment. Environment variables may be the same for all output destinations, or can be changed for individual print queues. Typical uses of environment variables include setting values that are used in all projects, like a company name, or using a different merge project depending on whether the destination is a printer, fax, email or archive server port.

Storage

Environment variables are stored in the EnvironmentVars.data file in the SystemResource folder. This file will be uploaded to the e-forms server when the project is transferred.

To create an environment variable

1. From the Main menu, select Tools > Environment Variables. The Environment Variables dialog will be displayed.
2. To create a new environment variable, click the Add Row button.
3. Type the name of the environment variable in the Name field.
4. Type the value of the environment variable in the Value field.
5. Click OK.
Job Ticket Variables

All the job ticket values passed through with the spool file, or added by the e-forms server are available wherever system variables are available in the design. The job ticket variables available to a project must be defined in the following configuration files found in the Win32User folder:

- Text, CSV and XML input files: HeaderParams.cfg
- JDE PDF input files: JDEHeaderParams.cfg

Any changes you make to configuration files on the design workstation will be transferred to the e-forms server with the project. Refer to Transferring Projects to the e-forms Server for more information.
EXPRESSIONS AND CONDITIONS

You can create expressions and conditions to combine and manipulate text, calculate values, summarize data in tables and test when objects, sections and pages should be displayed.

The following topics are covered in this chapter:

• Expressions
• Conditions
• Building Blocks
• Syntax and Order of Precedence Rules
Expressions

What do you want to know about?
- Building Expressions
- Using the Expression Builder
- Navigating and Editing Expressions

Building Expressions

Expressions are built using the Expression Builder dialog (referred to as the Expression Builder), which is opened whenever you click the button from either the Derived Variables dialog (input) or the User Variables dialog (output). The Expression Builder provides a range of operators and functions used to build expressions that evaluate values, called arguments, in a specific order, or syntax.

Expressions are evaluated from left to right subject to the Syntax and Order of Precedence Rules.

Using the Expression Builder

You must build an expression in a particular order. Generally you will follow the list boxes from left to right. The operators, functions and variables available to you are context sensitive. For example, when you select a variable with a numeric value, only functions that can process numbers will be available to you.

The following procedure explains how to build a simple expression to add the text “Inc.” to a derived variable called “Company_Name”.

To build the expression

1. From the Main menu, select Node>User Variables.
2. From the User Variables dialog, in the Variable Name field, type the name of the variable you want to create.
3. Click to open the Expression Builder.
4. From the Expression Builder, click concat in the Function section.
5 In the Variable section, expand and select the user variable.

6 Click on the Edit Value button and type the text you want to add to the user variable. The Expression Builder will display: concat(Company_Name, "Inc.").

7 Click OK to close the Expression Builder.

When you close the Expression Builder, you will be returned to the Derived Variables dialog where the expression will be displayed in the Value field, and the value the expression returns in the current section will be displayed in the Content field. The same procedure applies when building expressions from the User Variables dialog.

Navigating and Editing Expressions

- To move around the expression, use the arrow buttons, keyboard arrow keys and the cursor.
- To expand or reduce the selection, use the Level Up button or the Level Down button.
- To add text or number values, use the Edit Value button.
- To delete parts of an expression, select the part of the expression to be deleted and click the Delete button.
- To enclose parts of an expression in parentheses, select the part of the expression to be enclosed and click the Group button.

You can also enable direct editing of the expression in the dialog that calls the Expression Builder. Refer to General Preferences for more information.
Conditions

A condition is a special type of expression which produces a TRUE or FALSE value. Conditions allow you to vary the output of the project, depending on data within the input file. For more information about the ways you can use conditions, see the usage examples below.

Usage examples

Conditions can be used to determine:
- When input sections are created (Refer to Add Detail Sections and Add Input Footer Sections for more information).
- When sets are created (Refer to Defining Sets for more information).
- When defining derived values
- When calculating values or manipulating data in user variables

What do you want to know about?
- Building Conditions
- Using Expressions as Conditions

Building Conditions

The Condition Builder is used to build and edit expressions in much the same fashion as the Expression Builder. Refer to Expressions for more information. The Condition Builder can be opened from numerous dialogs, either by clicking the Edit button next to a condition field, or by clicking the button.

The following procedure explains how to build a simple condition to test whether the user variable called “Item_type” contains the word “new”.

To build a simple condition

1. From the Condition Builder, click contains in the Function section.
2. In the Variable section, expand and select the user variable.
3. Click on the Edit Value button and type “new”. The Condition Builder will display: contains(Item_type,"new")
4. Click OK to close the Condition Builder.
When you close the Condition Builder, you will be returned to the originating dialog where the expression will be displayed. To enable direct editing of the expression in the dialog that calls the Condition Builder, refer to General Preferences.

**Using Expressions as Conditions**

Any expression that evaluates to a non-zero value when true can be used as a condition. For example, the expression A-B is equivalent to the condition A!=B as both expressions return a zero value when A equals B and a non-zero value when A is not equal to B.
Building Blocks

The Expression Builder and Condition Builder dialogs contain list boxes containing the building blocks from which you can build an expression. The content of these dialogs is context sensitive; the building blocks displayed or available will depend on the task you are performing:

- Boolean Operators
- Operators
- Functions
- Variables
- System Conditions

Boolean Operators

Boolean operators are only available in the Condition Builder. The syntax of an expression involving boolean operators begins with the first argument, followed by the operator, and the second argument, separated by spaces:

condition1 OPERATOR condition2

Arguments can be other user defined conditions or system conditions. The following table describes the boolean operators available.
Other boolean operations can be performed by combining AND and OR Operators in expressions with the negation operator NOT; for example, a boolean difference operation is performed by the expression:

```
NOT condition1 AND condition2
```

### Operators

The usual syntax of an expression involving operators begins with the first argument, followed by the operator, and the second argument, with or without separating spaces:

```
argument1 OPERATOR argument2
```

The negation operators are the only exceptions to this rule, as they precede a single argument. Arguments can be numbers, text strings, variables or other functions or expressions. The following table describes the operators available.

---

**Table 9.1:**

<table>
<thead>
<tr>
<th>Boolean Operator</th>
<th>Meaning</th>
<th>Syntax and Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>logical AND</td>
<td>condition1 AND condition2</td>
</tr>
<tr>
<td></td>
<td>(intersection)</td>
<td>Returns TRUE if both condition1 and condition2 are true.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Example: contains(Price,&quot;.&quot;) AND contains(Price,&quot;0&quot;) returns TRUE if the data variable Price contains both &quot;,&quot; and &quot;0&quot;.</td>
</tr>
<tr>
<td>OR</td>
<td>logical OR</td>
<td>condition1 OR condition2</td>
</tr>
<tr>
<td></td>
<td>(union)</td>
<td>Returns TRUE if either condition1 or condition2 are true.</td>
</tr>
</tbody>
</table>
### Operator Summary

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Syntax and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>==</td>
<td>equal to</td>
<td>A==B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers or two strings and returns TRUE if equal. Case sensitive when comparing strings.</td>
</tr>
<tr>
<td>!=</td>
<td>not equal to</td>
<td>A!=B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers or two strings and returns TRUE if not equal. Case sensitive when comparing strings.</td>
</tr>
<tr>
<td>&gt;</td>
<td>greater than</td>
<td>A&gt;B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers and returns TRUE if A is greater than B.</td>
</tr>
<tr>
<td>&lt;</td>
<td>less than</td>
<td>A&lt;B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers and returns TRUE if A is less than B.</td>
</tr>
<tr>
<td>&gt;=</td>
<td>greater than or equal to</td>
<td>A&gt;=B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers and returns TRUE if A is greater than or equal to B.</td>
</tr>
<tr>
<td>&lt;=</td>
<td>less than or equal to</td>
<td>A&lt;=B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Compares the values of two numbers and returns TRUE if A is less than or equal to B.</td>
</tr>
<tr>
<td>*</td>
<td>*</td>
<td>A*B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Multiplies two numbers.</td>
</tr>
<tr>
<td>/</td>
<td>divide</td>
<td>A/B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Divides A by B.</td>
</tr>
<tr>
<td>+</td>
<td>add</td>
<td>A+B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Adds two numbers, or concatenates two strings.</td>
</tr>
<tr>
<td>-</td>
<td>subtract</td>
<td>A-B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Subtracts B from A.</td>
</tr>
<tr>
<td>NOT</td>
<td>logical negation</td>
<td>NOT condition</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reverses the logical value of an expression.</td>
</tr>
<tr>
<td>- (neg)</td>
<td>arithmetic negation</td>
<td>-B</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reverses the sign of a numeric value, i.e. -B=-1*B.</td>
</tr>
</tbody>
</table>

Refer to [Syntax and Order of Precedence Rules](#) for information regarding the evaluation of expressions using parentheses, multiple operators and mixed data types.
Functions

The syntax of a function begins with the function name, followed by an opening parenthesis, the arguments for the function separated by commas, and a closing parenthesis.

function(argument1,argument2,...)

Arguments can be numbers, text strings, variables, functions and expressions. The following tables describe the functions available.

Logical Functions

Logical functions can be used as conditions returning a TRUE or FALSE value. When used as arguments in expressions, logical functions return a value of 1 if true, and 0 if false.

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>changed</td>
<td>changed(name)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if the variable name is not equal to the value in the previous occurrence. Equivalent to the expression name!=previous(name).</td>
</tr>
<tr>
<td>contains</td>
<td>contains(string,find_text)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if string contains find_text. Example: contains(&quot;Heath;Adele;Mrs;;;OK;;;&quot;,&quot;OK&quot;) = TRUE</td>
</tr>
<tr>
<td>exists</td>
<td>exists(name)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if the variable name exists.</td>
</tr>
<tr>
<td>isfile</td>
<td>isfile(filename)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if the file filename exists.</td>
</tr>
<tr>
<td>isfolder</td>
<td>isfolder(foldername)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if the folder foldername exists.</td>
</tr>
<tr>
<td>isnumeric</td>
<td>isnumeric(name)</td>
</tr>
<tr>
<td></td>
<td>Returns TRUE if the variable name contains at least one numeral and any of the characters , . ( ) + - $ % or space. The test will fail if the variable contains any other character. Examples: isnumeric(&quot;$ 1,234.56&quot;) = TRUE isnumeric(&quot;$US1,234.56&quot;) = FALSE</td>
</tr>
<tr>
<td>Function</td>
<td>Syntax and usage</td>
</tr>
<tr>
<td>------------</td>
<td>------------------</td>
</tr>
<tr>
<td>aswords</td>
<td>aswords(number,option)</td>
</tr>
<tr>
<td></td>
<td>Converts a number into words. Specifying &quot;a&quot; in the second argument will generate a number phrase with “and” e.g. “Five hundred and twenty seven”</td>
</tr>
<tr>
<td></td>
<td>Examples:</td>
</tr>
<tr>
<td></td>
<td>aswords(&quot;123&quot;,&quot;&quot;) = &quot;One hundred twenty three&quot;</td>
</tr>
<tr>
<td></td>
<td>aswords(&quot;123&quot;,&quot;a&quot;) = &quot;One hundred and twenty three&quot;</td>
</tr>
<tr>
<td>at</td>
<td>at(row,column,length)</td>
</tr>
<tr>
<td></td>
<td>Returns a string of a specified length, starting at the specified row and column position in the current input page.</td>
</tr>
<tr>
<td></td>
<td>Example: if row 5 of the input page contains the letters of the alphabet in columns 1 to 26, then</td>
</tr>
<tr>
<td></td>
<td>at(5,12,3) = &quot;LMN&quot;</td>
</tr>
<tr>
<td>concat</td>
<td>concat(string1,string2)</td>
</tr>
<tr>
<td></td>
<td>Joins two text strings.</td>
</tr>
<tr>
<td></td>
<td>Example: concat(&quot;Adele &quot;, &quot;Heath&quot;) = &quot;Adele Heath&quot;</td>
</tr>
<tr>
<td>concatlist</td>
<td>concatlist(list_var,separator)</td>
</tr>
<tr>
<td></td>
<td>Returns a string containing all occurrences of the variable list_var separated by the character/string separator.</td>
</tr>
<tr>
<td>external</td>
<td>external(pathname,command_line)</td>
</tr>
<tr>
<td></td>
<td>Runs an external process using the specified pathname and command line.</td>
</tr>
<tr>
<td>find</td>
<td>find(string,find_text,start_char)</td>
</tr>
<tr>
<td></td>
<td>Finds one text string within another text string, starting from character position start_char, and returns the starting position. The first character in string is position 1. Returns 0 if find_text is not found in string.</td>
</tr>
<tr>
<td></td>
<td>Example: find(&quot;Mrs Adele Heath&quot;,&quot; &quot;,5) = 10</td>
</tr>
<tr>
<td>length</td>
<td>length(string)</td>
</tr>
<tr>
<td></td>
<td>Returns the length of a text string, including spaces.</td>
</tr>
<tr>
<td></td>
<td>Example: length(&quot;Mrs Adele Heath&quot;) = 15</td>
</tr>
<tr>
<td>lower</td>
<td>lower(string)</td>
</tr>
<tr>
<td></td>
<td>Converts a text string to lower case.</td>
</tr>
<tr>
<td>ltrim</td>
<td>ltrim(string)</td>
</tr>
<tr>
<td></td>
<td>Trims leading spaces from a text string.</td>
</tr>
<tr>
<td></td>
<td>Example: ltrim(&quot; left trim &quot;) = &quot;left trim &quot;</td>
</tr>
<tr>
<td>Function</td>
<td>Syntax and usage</td>
</tr>
<tr>
<td>-----------</td>
<td>------------------</td>
</tr>
<tr>
<td>mod</td>
<td><code>mod(number, base)</code>&lt;br&gt;Returns the remainder after number is divided by base.&lt;br&gt;Example: <code>mod(51, 7) = 2</code></td>
</tr>
<tr>
<td>num</td>
<td><code>num(string)</code>&lt;br&gt;Converts a text string into a number.&lt;br&gt;Examples:&lt;br&gt;<code>num(&quot;12ab3&quot;) = 123</code>&lt;br&gt;<code>num(&quot;18+9/3&quot;) = 1893</code>&lt;br&gt;<code>num(&quot;nine&quot;) = 0</code></td>
</tr>
<tr>
<td>pad</td>
<td><code>pad(string, length)</code>&lt;br&gt;If string is shorter than length, extends string to length with spaces.&lt;br&gt;If string is longer than length, truncates string to length.&lt;br&gt;Examples:&lt;br&gt;<code>pad(&quot;abc&quot;,&quot;6&quot;) = &quot;abc  &quot;</code>&lt;br&gt;<code>pad(&quot;abc&quot;,1) = &quot;a&quot;</code>&lt;br&gt;<code>pad(&quot;abc&quot;,-2) = &quot;abc&quot;</code>&lt;br&gt;<code>pad(&quot;abc&quot;,&quot;def&quot;) = &quot;&quot;</code></td>
</tr>
<tr>
<td>proper</td>
<td><code>proper(string)</code>&lt;br&gt;Capitalizes the first letter of each word in a text string.</td>
</tr>
<tr>
<td>previous</td>
<td><code>previous(var_name)</code>&lt;br&gt;Returns the value of variable var_name in the previous occurrence of the current section.</td>
</tr>
<tr>
<td>previousat</td>
<td><code>previousat(row, column, length)</code>&lt;br&gt;Returns a string of a specified length, starting at the specified row and column position in the previous input page.</td>
</tr>
</tbody>
</table>
### Function Syntax and usage

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>retrieve</strong></td>
<td>retrieve(string,ordinal_position,delimiter_char)</td>
</tr>
<tr>
<td></td>
<td>Extracts text from delimited fields in a text string. When the space character is used as a delimiter, leading spaces are ignored and sequential spaces are treated as a single delimiter.</td>
</tr>
<tr>
<td></td>
<td>Examples: retrieve(&quot;Abc;De;F;;Ghij;;&quot;,6,&quot;;&quot;) = &quot;Ghij&quot;</td>
</tr>
<tr>
<td></td>
<td>retrieve(&quot; Abc De F&quot;,2,&quot; &quot;) = &quot;De&quot;</td>
</tr>
<tr>
<td></td>
<td>Use the following arguments when specifying these delimiters:</td>
</tr>
<tr>
<td></td>
<td>ArgumentDelimiter</td>
</tr>
<tr>
<td></td>
<td>\new line</td>
</tr>
<tr>
<td></td>
<td>\tab</td>
</tr>
<tr>
<td></td>
<td>&quot;double quotation mark (&quot;)</td>
</tr>
<tr>
<td></td>
<td>\backslash ()</td>
</tr>
<tr>
<td></td>
<td>Example: retrieve(var,1,&quot;\t&quot;) returns the value before the first tab in the text variable var.</td>
</tr>
<tr>
<td><strong>rfind</strong></td>
<td>rfind(string,find_text,start_char)</td>
</tr>
<tr>
<td></td>
<td>Finds one text string within another text string by searching from the right to the left, starting from character position start_char, and returns the starting position. The first character in string is position 1. Position 0 specifies the last character in string. Returns 0 if find_text is not found in string.</td>
</tr>
<tr>
<td></td>
<td>Example: rfind(&quot;Heath,Adele,Mrs&quot;,&quot;&quot;,0) = 12</td>
</tr>
<tr>
<td><strong>round</strong></td>
<td>round(number,num_digits)</td>
</tr>
<tr>
<td></td>
<td>Rounds a number to a specified number of decimal places.</td>
</tr>
<tr>
<td></td>
<td>Examples: round(1234.567,2) = 1234.57</td>
</tr>
<tr>
<td></td>
<td>round(1234.56,0) = 1235</td>
</tr>
<tr>
<td></td>
<td>round(1234.56,-2) = 1200</td>
</tr>
<tr>
<td><strong>rtrim</strong></td>
<td>rtrim(string)</td>
</tr>
<tr>
<td></td>
<td>Trims trailing spaces from a text string.</td>
</tr>
<tr>
<td></td>
<td>Example: rtrim(&quot; right trim &quot;) = &quot; right trim&quot;</td>
</tr>
<tr>
<td><strong>sentence</strong></td>
<td>sentence(string)</td>
</tr>
<tr>
<td></td>
<td>Converts a text string to sentence case.</td>
</tr>
</tbody>
</table>
Refer to the Syntax and Order of Precedence Rules for information regarding the evaluation of nested functions.

**Date Functions**

The result of a date function depends on the current date order setting. Refer to Set Input Data Formats for more information.

<table>
<thead>
<tr>
<th>Function</th>
<th>Syntax and usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>date(year,month,day)</td>
</tr>
<tr>
<td></td>
<td>Returns a date value from arguments representing the year, month and day. Values of day and month outside the normal range are treated on the basis that month=0 is the last month of the previous year, and day=0 is the last day of the previous month.</td>
</tr>
<tr>
<td></td>
<td>The following examples assume a date order setting of d-m-y:</td>
</tr>
<tr>
<td></td>
<td>date(&quot;2003&quot;,&quot;4&quot;,&quot;19&quot;) = 19-4-2003</td>
</tr>
<tr>
<td></td>
<td>date(2000,0,0) = 30-11-1999</td>
</tr>
<tr>
<td></td>
<td>date(2004,-1,32) = 2-12-2003</td>
</tr>
<tr>
<td>day</td>
<td>day(date)</td>
</tr>
<tr>
<td></td>
<td>Returns the day of the month (a number between 1 and 31) from a date value.</td>
</tr>
<tr>
<td></td>
<td>Example: day(&quot;19/4/03&quot;) = 19 when date order is d-m-y</td>
</tr>
</tbody>
</table>
Any non-numeric parsing character is permitted in a string representing a date value, for example, the strings "19 4 03" and "19.04.2003" will return the same value when used as arguments in a date function.

Refer to Syntax and Order of Precedence Rules for information on using the addition and subtraction operators with date values.

**Variables**

The types of variables that can be used in conditions as arguments of operators and functions are summarized below:

<table>
<thead>
<tr>
<th>Variable Type</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>data variable</td>
<td>input data (refer to Design Principles for more information)</td>
</tr>
<tr>
<td>user variable</td>
<td>user defined expressions relating to section data (refer to User Variables for more information)</td>
</tr>
<tr>
<td>global user variable</td>
<td>user defined expressions relating to other sections or the whole project (refer to Global User Variables for more information)</td>
</tr>
<tr>
<td>system variable</td>
<td>system supplied data (refer to System Variables for more information)</td>
</tr>
<tr>
<td>lookup variable</td>
<td>data from an external database (refer to Lookup Variables for more information)</td>
</tr>
<tr>
<td>environment variable</td>
<td>data that applies to all projects (refer to Environment Variables for more information)</td>
</tr>
<tr>
<td>job ticket variable</td>
<td>job file attributes (refer to Job Ticket Variables for more information)</td>
</tr>
</tbody>
</table>
The variables that are available to you at any time, will depend on the type of expression you are building, and where you are building it from. User variables and data variables are available only in the section they are defined in.

**System Conditions**

When working with text an JDE PDF input files, Create!form Director provides pre-defined conditions that you can use to:

- Conditionally apply an input section
- Conditionally define derived variables

System conditions will only be displayed in the Condition Builder when you are performing one of the above tasks. The following table describes the system conditions available:

<table>
<thead>
<tr>
<th>Pre-defined Conditions</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sys.FirstPageOfDoc</td>
<td>Returns TRUE when the current section is on the first page of the project.</td>
</tr>
<tr>
<td>Sys.ContPageOfDoc</td>
<td>Returns TRUE when the current section is not on the first page or the last page of the project.</td>
</tr>
<tr>
<td>Sys.LastPageOfDoc</td>
<td>Returns TRUE when the current section is on the last page of the project.</td>
</tr>
<tr>
<td>Sys.FirstPageOfSet</td>
<td>Returns TRUE when the current section is on the first page of the set for the project.</td>
</tr>
<tr>
<td>Sys.ContPageOfSet</td>
<td>Returns TRUE when the current section is not on the first page or the last page of the set for the project.</td>
</tr>
<tr>
<td>Sys.LastPageOfSet</td>
<td>Returns TRUE when the current section is on the last page of the set for the project.</td>
</tr>
</tbody>
</table>
Syntax and Order of Precedence Rules

Expressions are evaluated using a specific set of rules, or syntax, that depends on the operators and functions you use in the expression. The Condition Builder and Expression Builder dialogs will assist you in building expressions with a valid syntax; however, you must follow a few basic rules to ensure the expression will perform the way you expect:

- Mixed Data Types
- Addition Operator
- Subtraction Operator
- Comparing Text Strings
- Order of Precedence
- Nested Functions
- Parsing
- Names of Variables

Mixed Data Types

Avoid mixing text, numeric, and date values in the same expression as this can produce unexpected results. All variable data is stored as a string, but will be interpreted as a number in an arithmetic operation. For example:

"-2"*'3" evaluates to the number -6
2*'3A4" evaluates to the number 68
2*'3-4" evaluates to the number 68
2*'3A4-" evaluates to the number -68
"two"*'three" evaluates to the number 0
Addition Operator

The addition operator (+) is a special case. The rules governing the addition of numeric, text and date arguments are:

• If both arguments of an addition are numeric, an addition is performed. For example: 1+1 evaluates to the number 2.

• If either argument of an addition is a string, the arguments are concatenated. For example: 1+'1' evaluates to the text string "11".

• If one argument is a date and the other is a number, a date addition is performed. For example: if Today is a date variable (m-d-y) with a value of 9-6-2004, the expression Today+1 evaluates to the date value 9-7-2004, whereas the expression "9/6/2004"+1 evaluates to the string "9/6/20041".

Subtraction Operator

The subtraction operator (-) is a special case. The rules governing the subtraction of numeric, text and date arguments are:

• If both arguments of a subtraction are either numeric or text, a subtraction is performed. For example: "a5"-2 evaluates to the number 3.

• If the first argument is a date and the second is a number, a date subtraction is performed. For example: if Today is a date variable (m-d-y) with a value of 9-6-2004, the expression today-1 evaluates to the date value 9-5-2004, whereas the expression "9/6/2004"-1 evaluates to the number 962003.

• If both arguments are dates, a date difference is performed. For example: Sys.Today-InvDate evaluates to the number of days difference between today’s date and the date variable InvDate, whereas the expression "9/6/2004"."9/6/2003" evaluates to the number 1.

Comparing Text Strings

Avoid using the comparison operators <=, >=, <, > to compare text strings as this can produce unexpected results. The rules governing comparisons of numeric, text and date arguments are:

• If either argument of a comparison is a number, a numeric comparison is performed. For example: "10">2 returns TRUE.
• If both arguments are strings, a string comparison is performed based on the ASCII collating sequence, and is case sensitive. For example: "10">"2" returns FALSE.
• If both arguments of a comparison are dates, a date comparison is performed.

**Order of Precedence**

Expressions are evaluated in the following order of precedence:
- operations inside parentheses ( ) are evaluated first, followed by
- - (negation)
- *, /
- +, -\_
- ==, !=, <=, >=, <, >
- NOT
- AND, OR

Operators with the same order of precedence e.g. + and -, are evaluated from left to right. For example, the expression 3*2+6/-2 evaluates to 3, whereas 3*(2+6/-2) evaluates to -3.

**Nested Functions**

Functions can be used as arguments for other functions, and can be nested. For example, if the variable A has a value of "Andersch,Jorge", then the expression:

```
length(substring(A,find(A","",1)+1,length(A)-find(A","",1)))
```

will return the value 5 (the length of the substring "Jorge").

**Parsing**

Certain characters or combinations of characters, when included in text strings, can have specific meanings in an expression:
Expressions and Conditions

Syntax and Order of Precedence Rules

Create!form Director

For example,
"the literal " is ignored" evaluates to the literal " is ignored
"C:\\Program Files\\Create!form" evaluates to C:\Program Files\Create!form
2*"1A2B3-" evaluates to -246
2*"1A2-3" evaluates to 246

Names of Variables

Avoid using function names as names of variables as this can produce unexpected results. Refer to Functions for more information.
206* EXPRESSIONS AND CONDITIONS

- Syntax and Order of Precedence Rules
You can customize the look and feel of the Create!form Director interface. The following topics are covered in this chapter:

- About User Preferences
- General Preferences
- Design Window Preferences
- Input Window Preferences
About User Preferences

You can customize the look, feel and behavior of Create!form Director to suit the way you work. User preference settings are specific to the local copy of Create!form Director; changes made to user preferences are not saved with the project.

All user preferences are set from the Preferences dialog, which will be displayed by selecting Tools>Prefereces from the Main menu. The categories of settings available are displayed in the tree view on the left of the dialog. Select the category in the tree view to display the corresponding controls on the right.

* General Preferences
* Section Borders
* Input Source Preferences

![Preferences dialog](image-url)
• Design Window Preferences
• Design Tree
• CSV
• JDE PDF
• JDE PDF Sections
• Text
• Text Sections
• Design Tree
General Preferences

Select Tools>PREFERENCES from the Main menu to display the Preferences dialog and select General in the tree view.

The following general preferences settings are available:

**Default Project Directory**
Click the button to change the Project Directory.

**Common Project Directory**
Click the button to change the Common Project Directory.

**Allow Direct Editing of Expressions and Conditions**
Select the checkbox to enable direct editing of expressions and conditions in dialogs without opening the Expression Builder or the Condition Builder.

**Make Backup on Save**
Select the checkbox to make a backup copy of the project and DataMap files when the project is saved. Refer to **Saving a Project** for more information.

To use a backup version of a project file and a DataMap file, you must manually change the name of the backup files to the correct filename extension. Refer to **Managing Project Files** for more information.

**Restore Suppressed Messages**
Select this option to restore messages that have previously been suppressed. A message is suppressed when the “Do not show this message again” option is selected on a message dialog.

Section Borders

Select Tools>PREFERENCES from the Main menu to display the Preferences dialog and select General>Section Borders in the tree view. The following preferences settings are available:

- The color of a section when it is selected
- The color of the corresponding section in an inactive window
- The thickness of the border
Input Source Preferences

Select Tools>Preferences from the Main menu to display the Preferences dialog and select General>Input Source in the tree view. The following preferences settings are available:

Display Font

- Select the input file source view display font in the drop-down.
- Select the font Size.
- Select the Monospace only checkbox to restrict the fonts available in the font drop-down.

Colors

Select the colors for the text and the background.

Scroll bars

Display and hide the horizontal and vertical scroll bars.
Design Window Preferences

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Design in the tree view. The following preferences settings are available:

**Scale Factor**
Sets the default view magnification.

**Background Color**
Select the color for the window background.

**Show Rulers**
Horizontal and vertical rulers are available to assist in node layout and placement.

**Show Tooltips**
Select the checkbox to display tooltips when the cursor is moved over nodes.

Design Tree

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Design>Tree in the tree view. The following preferences settings are available:

**Node Text**
Select an option to annotate nodes with either a description, or the properties of the node.

**Colors**
Select the annotation text colors.

**Node Images**
Select a node type from the drop-down, and click the Browse button () to display the Select Image dialog from which you can change the image that will be displayed at all nodes of that type. A selection of images are provided, but you can also add your own icons. The controls on the Select Image dialog will allow you to add and delete image files.

To highlight invalid nodes, select the Mark invalid nodes checkbox.
**Connectors**
Select the line type, style and color of node connectors.

**Selected node frame color**
Select the color of the frame that outlines the selected node.
Input Window Preferences

CSV

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (CSV) in the tree view. The following preferences settings are available:

Input file size

To limit the size of an input data file to be used as a sample, select the Limit to checkbox and enter the maximum number of lines you want to use. To most accurately represent the sample data, it is recommended that you don’t limit the number of lines unless the number of lines affects the processing speed. Whenever you open a project where the input file size exceeds the limit, you will be prompted to truncate the input file.

Header

- Select the Show checkbox to display the CSV input file header in the Input Window.
- To change the background color, click the Background color field to display the Color Picker dialog.

Detail

- Select the Show row numbers checkbox to display the row number in the Input Window.
- To alternate the background color between consecutive sets, select different colors for odd and even sets.

JDE PDF

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (JDE PDF) in the tree view. The following preferences settings are available:

View

To set the default page view magnification, choose Fit to Width, Fit in Window or Actual Size from the View drop-down list.
Input File Size

To limit the size of an input data file to be used as a sample, select the Limit to checkbox and enter the maximum number of lines you want to use. To most accurately represent the sample data, it is recommended that you don’t limit the number of lines unless the number of lines affects the processing speed.

Whenever you open a project where the input file size exceeds the limit, you will be prompted to truncate the input file.

Show Rulers

Horizontal and vertical rulers are available to measure column (character) and row (line) positions on the input page.

Enable Tooltips

Select the checkbox to display names as the cursor moves over data variables in the Input Window.

JDE PDF Sections

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (JDE PDF)>Sections in the tree view. The following preferences settings are available:

Section borders

- Select the Show checkbox to display section borders in the Input Window.
- Select the section border line style from the Style drop-down.

Selected section

- Select the Highlight background checkbox to display the selected section in the background color shown. To change the background color, click the background color field to display the Color Picker dialog.
- To change the variable background color, click the Variable background color field to display the Color Picker dialog.
- To change the selected variable frame color, click the Selected variable frame color field to display the Color Picker dialog.
Unselected Sections
Select the Highlight background checkbox to display unselected sections in the background color shown. To change the background color, click the background color field to display the Color Picker dialog.

Text
Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (Text) in the tree view. The following preferences settings are available:

View
To set the default page view magnification, choose Fit to Width, Fit in Window or Actual Size from the View drop-down list.

Input file size
To limit the size of an input data file to be used as a sample, select the Limit to checkbox and enter the maximum number of lines you want to use. To most accurately represent the sample data, it is recommended that you don’t limit the number of lines unless the number of lines affects the processing speed.
Whenever you open a project where the input file size exceeds the limit, you will be prompted to truncate the input file.

Grid
A grid can be displayed in the Input Window to assist in mapping sections and data variables. The grid lines correspond to the column and row numbering displayed on the rulers.
Use the Grid options to:
• Hide and show a grid
• Show grid as points
• Change the grid line style

Rulers
Horizontal and vertical rulers are available to measure column (character) and row (line) positions on the input page. Use the Rulers options to:
• Hide or show ruler.
• Change the cursor indicator color.
Display font

• Select the Monospace only checkbox to restrict the fonts available in the font drop-down.

• Select the input file display font in the drop-down.

Enable Tooltips

Select the checkbox to display names as the cursor moves over data variables in the Input Window.

Text Sections

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (Text)>Sections in the tree view. The following preferences settings are available:

Section borders

• Select the Show checkbox to display section borders in the Input Window.

• Select the section border line style from the Style drop-down.

Selected section

• Select the Highlight background checkbox to display the selected section in the background color shown. To change the background color, click the background color field to display the Color Picker dialog.

• To change the variable background color, click the Variable background color field to display the Color Picker dialog.

• To change the selected variable frame color, click the Selected variable frame color field to display the Color Picker dialog.

Unselected Sections

• Select the Highlight background checkbox to display unselected sections in the background color shown. To change the background color, click the background color field to display the Color Picker dialog.

• To change the variable background color, click the Variable background color field to display the Color Picker dialog.
XML

Select Tools>Preferences from the Main menu to display the Preferences dialog and select Input (XML) in the tree view. The following preferences settings are available:

**Tooltips**
You can choose to display tooltips when the cursor hovers over an XML element, and optionally, the tooltip will display values, variable and section names, and the type of node.

**Initial document view**
When the document is opened, choose if the whole tree is expanded, or only the first level.

**Colors**
Select a component of the tree to display the current color setting. To customize colors, clear the Use system color checkbox and click the color field to display the Color Picker dialog.
GLOSSARY

This appendix provides a list of commonly used Create!form terms.
# Glossary

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>argument</td>
<td>The component of a function or expression that defines a value.</td>
</tr>
<tr>
<td>common project directory</td>
<td>The directory used to store shared project resources.</td>
</tr>
<tr>
<td>CommonProject directory</td>
<td>The default common project directory</td>
</tr>
<tr>
<td>condition</td>
<td>A user-defined logical test that results in either a true or false value. Conditions enable the project to behave differently according to the content of the input file or the production environment.</td>
</tr>
<tr>
<td>data section</td>
<td>A section that is brought into the project from the input file.</td>
</tr>
<tr>
<td>DataMap</td>
<td>A DataMap records how the different parts of an input file are identified and labelled.</td>
</tr>
<tr>
<td>data variable</td>
<td>A variable defined in the input file.</td>
</tr>
<tr>
<td>derived variable</td>
<td>A user-defined expression, defined in the input.</td>
</tr>
<tr>
<td>e-forms server</td>
<td>The system that hosts Create!form Server, which merges the project with incoming spool files.</td>
</tr>
<tr>
<td>enterprise server</td>
<td>ERP system or other application that is originator of input files.</td>
</tr>
<tr>
<td>expression</td>
<td>A user-defined statement that performs calculations and derives values from the input data.</td>
</tr>
<tr>
<td>form project</td>
<td>See project.</td>
</tr>
<tr>
<td>function</td>
<td>A pre-defined formula that performs a calculation from user defined arguments.</td>
</tr>
<tr>
<td>global user variables</td>
<td>A user-defined variable, global to the project.</td>
</tr>
<tr>
<td>input file</td>
<td>A sample file from the enterprise server, used to build the DataMap.</td>
</tr>
<tr>
<td>lookup variable</td>
<td>A variable that derives its value from an external database.</td>
</tr>
<tr>
<td>operator</td>
<td>A component of an expression or condition that defines an arithmetic, logical or text operation on adjacent arguments.</td>
</tr>
<tr>
<td>project</td>
<td>A project is a record of the way you want to redesign the input file. Also referred to as a form project.</td>
</tr>
<tr>
<td>project directory</td>
<td>The directory where the project resources are stored.</td>
</tr>
<tr>
<td>repagination</td>
<td>The process of expanding or condensing pages in the project depending on the space required.</td>
</tr>
<tr>
<td>Term</td>
<td>Definition</td>
</tr>
<tr>
<td>------------------</td>
<td>---------------------------------------------------------------------------</td>
</tr>
<tr>
<td>set</td>
<td>A user-defined construct that divides the input file into its basic</td>
</tr>
<tr>
<td></td>
<td>document units, that the project is designing</td>
</tr>
<tr>
<td>subform</td>
<td>A subform is a user defined graphical object</td>
</tr>
<tr>
<td>system variable</td>
<td>System variables are predefined and relate to the project. They</td>
</tr>
<tr>
<td></td>
<td>include such items as the number of pages in the set or document.</td>
</tr>
<tr>
<td>template</td>
<td>Defines the parameters in the job ticket.</td>
</tr>
<tr>
<td>user variable</td>
<td>User variables are user-defined expressions that relate only to the</td>
</tr>
<tr>
<td></td>
<td>section they were created in.</td>
</tr>
</tbody>
</table>
INDEX

A
Addition Operator 203
Appendix A
  Glossary 219
Archiving 176

B
Before You Start 52
Boolean Operators 192
Building Blocks 192

C
Changing the Input Data Format 69, 112
Chapter Five
  Input Design 51
Chapter Four
  Viewing and Navigating 43
Chapter Three
  Getting Started 25
Common Project Directory 27
Comparing Text Strings 203
Conditions 190
  Building 190
  Using Expressions and Conditions 191
Create!form Components 17
Create!form Director Features 11
Create!form Director Requirements 12
Create!form Workflow 16
Creating Derived Variables 118
CSV File Structure 62
c t 41
Customizing
  About User Preferences 208
  Text 216
D
Data Variables 55, 67, 83, 110
  Changing Properties 111
  Changing the Properties of Columns 69
Create 84
Create in a Detail Section 86
Expand 87
In Detail Sections 69
In the Input Header Section 68
Data Variables, Derived Variables, and Section Variables 179
Databases 170
DataMap
  Edit Shared 59
  Embed 60
  Use Shared 60
DataMap Options 31
DataMap Options When Creating a New Project 58
DataMaps 57
  Embedded and Shared 57
  Options while Designing a Project 58
Defining Destinations 163
Defining Destinations by Address 165
Defining Destinations by Expression 168
Defining Login Details 165
Defining Named Destinations 167
Defining Sections 67
Defining Sets 65, 81
Derived Variables 56
Design Flowchart 74
Design Flowcharts 33
Design Principles 73
Design Tree 212
Design Window 45
Design Window Preferences 212
Designing for CSV Input Files 62
Designing for JDE PDF Input Files 113
Designing for Text Input Files 70
Designing from the Top-Down 73
Detail Section
   Rules for Creating Multiples 102
Detail Sections 98
   Adding 100
   Expanding 101
Duplicating 161

E
Edit an Embedded DataMap 59
Email 176
Environment Variables 185
Expressions 188

F
Fax 176
Filtering 157
Functions 195
   Logical 195

G
General Preferences 210
Get File Node 147
Getting Help 13
Getting Started
   What is a Project 26
Global User Variables 181
Glossary 220

I
Input File
   Jumping Directly to a Set or Page 48
   Moving Between Sections 49
   Navigating 48

Create!form Director
Selecting 52
Settings 78
Viewing the Source 53
Input File Header Section 92
Input File Properties 64, 107, 114
Input File Source View 46
Input File Type 31
Input Files
Examining how Sections are Populated 73
Input Footer Sections 102
Adding 103
Creating Multiples 104
Input Header Section
Expanding 94
Input Options 152
Input Sections 72
Common Tasks 88
Deleting 92
Multiple 91
Input Source Preferences 211
Input Tree 45
Input Window 45
Input Window Preferences 214
CSV 214
JDE PDF 214
Text Sections 217
XML 218
Installation Requirements 22
Installing Create!form Director 23

J
JDE PDF Sections 215
Job Ticket Variables 186

K
Key Design Concepts 120
Key Input Design Concepts 54
Key Tasks 153

L
Lookup Variables 183

M
Merge Configurations 162
Merging 162
Mixed Data Types 202
Multiple Input Headers Sections
   Rules for Creating 98

N
Names of Variables 205
Naming Destination Documents 165
Navigating and Editing Expressions 189
Nested Functions 204
New Project Wizard 29
Node
   Delete File 148
   Run Program 149
Node Properties 136
Nodes 122
   Adding 128
   Copying and Pasting 133
   Deleting 135
   Edit 132
   Moving 133
   Parent, Sibling and Child Relationships 124
   Selecting and Editing 130
   Sequence and Timing 126
   Using Variables and Expressions to Control Actions 126
O
Operators 193
Order of Precedence 204
Output Options 174
    Output Formats 174
Output to File 175

P
Page Size
    Calculation 78
Parsing 204
Printing 175
Printing and Previewing Projects 38
Product Overview 16
Project
    Creating 29
    Managing Files 41
    Opening 37
    Packing 39
    Publishing 170
    Saving 36
Project Directories 27
Project Directory 27
Project Type 31

R
Recursion 175

S
Section Borders 210
Sections 55
    Defining 110
Send Node 136
Sets 55
    Defining 108, 115
Setting the Input Page Size 76