Data Definition (DefTool) Guide for Oracle Siebel eStatement Manager

Version 4.7

May 31, 2007

ORACLE®

Contents

1 Preface

About Customer Self-Service and eaSuite™ 9

About This Guide 10

Related Documentation 10

2 Introduction to eStatement Manager Applications

eStatement Manager Applications 11

What is Electronic Statement Presentment? 11

The Statement Mastering Process 11

What is an Application? 12

What is a View? 12

How Do I Create an Application? 12

- What is a DDF? 13
- What is an ALF? 14
- The HTML Template 14

What is DefTool? 15

What is Composer? 15

Using the National Wireless Sample Application 15

After Creating an eStatement Manager Application 16

3 Overview of Creating a DDF with DefTool

Before Creating Your DDF 17

Steps for Creating a DDF 17

The DefTool Interface 18

- The DDF Tree 19
- The DefTool Work Area (Search Area) 19
- Menus 20

Toolbars and Icons 27

Cutting, Copying, and Pasting 30

4 Creating a New Data Definition File

Overview 35

Creating a DDF for DJDE Files 35

Creating a DDF for AFP Files 41

AFP Font Settings 46

Creating a DDF for DB Extract Files 47

Creating a DDF for Line Printer Format Files 50

Creating a DDF for Metacode Files 52

Metacode Settings 54

5 Writing Data Extraction Rules

What are Data Extraction Rules? 57

Identifying Unique Properties of Data to be Extracted 57

Search Area (Location) 57

Data Patterns (Content Search) 59

Field Properties 60

Marker Properties 61

Table Properties 61

Creating an Extraction Rule (Overview) 62

Document Style (Start of a New Document) 63

Document Recipient (Primary Key) 68

Creating Additional Page Styles 72

Overview of Fields and Field Anchors 73

Creating a Field Definition 74

6 Custom Post-Conversions

Using Custom Post-Conversions 77

Creating a New Custom Post-Conversion 78

Editing a Post-Conversion 81

Removing a Custom Post-Conversion from a Field or Column 81

Deleting a Custom Post-Conversion from an Application 82

Sharing Custom Post-Conversions with other Applications 82

Date and Time Format Constraints 84

Examples of Custom Post-Conversions 84

7 About Field Anchors

What are Field Anchors? 89

Creating a Field Anchor 92

Editing an Anchor 94

Deleting an Anchor 95

8 Tables

When to Use Tables 97

Extracting Tables for Text Presentation 97

Extracting Tables for Chart Presentation 97

Table Definitions 98

Creating a Table 99

Defining a Table in National Wireless 99

Defining a Table Start Marker 100

Defining a Table End Marker 103

Defining Table Columns 107

Defining the Table Anchor 109

9 Defining Groups

When to Use Groups 111

Identifying the Components of a Group Pattern 112

Table Columns 113

Creating a Group 114

Defining a Group Start Marker 115

Defining a Group End Marker 119

Defining Group Properties 122

10 Presenting Sub-Account Data
When to Use Sub-Account Indexing 125
Sample Sub-Document Indexing Application 125
Setting up Sub-Account Indexing 126
Defining Sub-Account Groups, Subkey, and Group Fields in a DDF 127
Adding a Group Field 130
Modifying a Group Field or Subkey Properties 131
Deleting a Group Field or Subkey 131

11 Using the Extraction Simulator
Using Simulator for the First Time 133
Navigating in the Simulator 133
Simulating Field Extractions 135
Simulating Table Extractions 136
Simulating Selective Extraction 137

12 Regular Expressions
What is a Regular Expression 141
Purpose of Regular Expressions 141
Using Regular Expressions with eStatement Manager 142
Writing Regular Expression Patterns 142
Illustration 1: Shorthand Date Formats MM/DD/YYYY 142
Illustration 2: Varied Shorthand Date Formats 144
Illustration 3: Currency Values 145
Illustration 4: Longhand Date Formats 146
Illustration 5: Data Values that Vary Widely in Pattern 147
Illustration 6: Static data strings 148
Illustration 7: Partially Static Data Strings 149
Regular Expressions Reference Chart 150

13 Dynamic Pattern Matching
What is a Dynamic Field? 153

Setting Dynamic Pattern Markers in DefTool 153

Simulating Dynamic Field Values 157

Completing the Dynamic Pattern Link in Composer 158

14 Element IDs

Element IDs 165

15 Appendix A: Data Definitions

EBNF Description 169

Data Definitions 169

Index

1 Preface

About Customer Self-Service and eaSuite™

Oracle has developed the industry's most comprehensive software and services for deploying Customer Self-Service solutions. **eaSuite**™ combines electronic presentment and payment (EPP), order management, knowledge management, personalization and application integration technologies to create an integrated, natural starting point for all customer service issues. eaSuite's unique architecture leverages and preserves existing infrastructure and data, and offers unparalleled scalability for the most demanding applications. With deployments across the healthcare, financial services, energy, retail, and communications industries, and the public sector, eaSuite powers some of the world's largest and most demanding customer self-service applications. eaSuite is a standards-based, feature rich, and highly scalable platform, that delivers the lowest total cost of ownership of any self-service solution available.

eaSuite consists of four product families:

- Electronic Presentment and Payment (EPP) Applications
- Advanced Interactivity Applications
- Enterprise Productivity Applications
- Development Tools

Electronic Presentment and Payment (EPP) Applications are the foundation of Oracle's Customer Self-Service solution. They provide the core integration infrastructure between organizations' backend transactional systems and end users, as well as rich e-billing, e-invoicing and e-statement functionality. Designed to meet the rigorous demands of the most technologically advanced organizations, these applications power Customer Self-Service by managing transactional data and by enabling payments and account distribution.

- **eStatement Manager**™ is the core infrastructure of enterprise Customer Self-Service solutions for organizations large and small with special emphasis on meeting the needs of organizations with large numbers of customers, high data volumes and extensive integration with systems and business processes across the enterprise. Organizations use eStatement Manager with its data access layer, composition engine, and security, enrollment and logging framework to power complex Customer Self-Service applications.
- **ePayment Manager**™ is the electronic payment solution that decreases payment processing costs, accelerates receivables and improves operational efficiency. ePayment Manager is a complete payment scheduling and warehousing system with real-time and batch connections to payment gateways for Automated Clearing House (ACH) and credit card payments, and payments via various payment processing service providers.

Oracle's **Development Tools** are visual development environments for designing and configuring Oracle's Customer Self-Service solutions. The Configuration Tools encompass data and rules management, workflow authoring, systems integration, and a software development kit that makes it easy to create customer and employee-facing self-service applications leveraging eaSuite.

About This Guide

This guide describes the tasks required to design and create an eStatement application for presenting statements online. It provides instructions on how to use the eStatement DefTool design and development tool.

This guide is intended for the application developer and those involved in the process of designing an eStatement application for systems with DJDE, AFP, DB Extract, line printer, or Metacode data input files.

This guide assumes you have:

- Completed an eStatement Mastering Plan.
- Installed the eStatement tools, DefTool and Composer.

Related Documentation

The following online Help is available in the eStatement Command Center:

Online	How to Access
DefTool and Composer User Help	In DefTool or Composer, select `Help>Help Topics`.

This guide is part of the eStatement Manager documentation set. For more information about implementing your eStatement Manager application, see one of the following guides:

Print Document	Description
Installation Guide for Oracle Siebel eStatement Manager	How to install eStatement Manager and configure it in a distributed environment.
Presentation Design (Composer) Guide for Oracle Siebel eStatement Manager	How to use Composer to define the rules for mapping data to templates for viewing statements.
Administration Guide for Oracle Siebel eStatement Manager	How to set up and run a live eStatement Manager application in a J2EE environment.
Deploying and Customizing J2EE Applications Guide for Oracle Siebel eStatement Manager	How to deploy and customize the J2EE applications provided by eStatement Manager. This guide also describes how to deploy the Sample application provided by eStatement Manager and how to validate that it is set up correctly by running a job through your installed eStatement Manager environment.

2 Introduction to eStatement Manager Applications

eStatement Manager Applications

eStatement Manager® is a high-performance enterprise Web statement and customer management solution that provides re-purposing and delivery of information over the Web. eStatement Manager is dependent on a number of distributable components, services, and web-based rapid application development (RAD) tools.

eStatement Manager provides a set graphical design tools for application development, called DefTool and Composer, and a platform-independent web tool called the Command Center that provides centralized control of the activity in the production environment.

What is Electronic Statement Presentment?

Electronic statement presentment (ESP) allows customers to view their financial statements on the Web. The primary functions of electronic statement presentation applications are: triggering email notifications, time-sensitive sticky content, and a targeted use of web page design, advertising, incentives and messaging.

The Statement Mastering Process

Statement Mastering is the process of determining your organization's online presentment requirements and defining the eStatement Manager application needed to enable your users to view statements on the Web.

During the mastering process, the development team analyzes printed copies of each bill type, along with a representative production file, and identifies the online statements your custom application must present. This process identifies the data, text, and graphics your statements must bring together, along with any conditional business logic required.

The result of Statement Mastering is a Requirements Definition Document defining the custom views and design files your application needs.

Use the Requirements Definition Document along with the instructions in this guide to create an application for your organization. This guide describe how to use the eStatement Manager DefTool to specify how eStatement Manager can extract data from your data input file (the data extraction rules), and Composer to define how data and other elements map to HTML templates for Web presentment.

To make an ESP system, statements must be converted from a normal print format to an electronic presentation. A printed file originates from a data source. The resulting data stream from the source is formatted for printing functions. eStatement Manager also uses the data source, structuring the data by using set rules, then displaying the data on a pre-configured HTML template. There can be several different sets of rules and templates, and the templates can also contain business rules to modify the type of marketing and customization that is displayed. The end result is the same account information provided in a print statement is also provided in an electronic presentation over the Web.

What is an Application?

An eStatement Manager Application is a customized set of eStatement Manager files (DDFs, ALFs, statement HTML templates, and group HTML templates) created exclusively to extract and present the statements of a service provider online. You must design a custom application with unique DDF, ALF, and HTML files for each input data source. Each application has a unique name. A single service provider can have more than one application if they need to present data from input data sources with different formats or content.

What is a View?

A view is a set of design files that result in a particular presentation of statement data. The view files enable a user to dynamically display formatted statements live on the Web, receive email notifying them that an online statement is available, or to view account data in a static format online.

An eStatement Manager application can have multiple statement views for presenting different levels of information such as a summary page and statement detail pages. Dynamic Web views can present statement data in HTML, XML, CSV, or Chart format.

A typical HTML-based statement view (dynamic or static) consists of a pair of DDF and ALFs, and one or more associated HTML templates. These files identify which data to extract from the input file and how to display it to the user.

Refer to the Requirements Definition Document for your organization. This document resulted from the Statement Mastering process and defines each view you need and the design files you must create for each view.

How Do I Create an Application?

Defining an eStatement Manager application requires two phases:

- Defining data extraction
- Presentation design

Defining data extraction consists of:

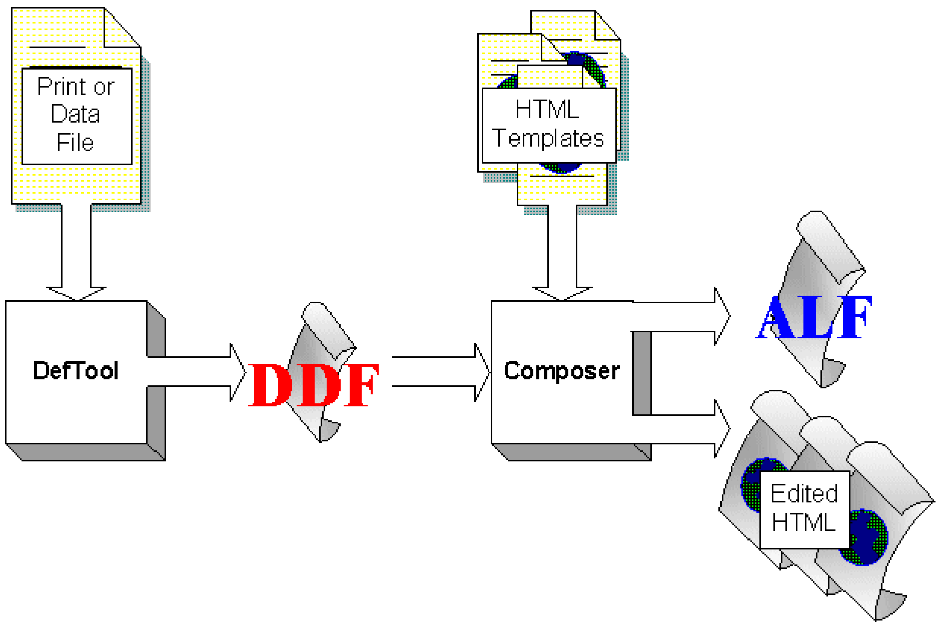
- Defining the file format properties of the input data file.
- Using the eStatement Manager DefTool to analyze and define the rules for extracting the data fields, tables, and groups from the input data source.

Presentation design consists of:

- Creating the HTML templates used to compose the statements. (Use the tool of your choice to create HTML templates.)

- Using the eStatement Manager Composer to map the fields, tables, and groups (defined using DefTool) to the HTML templates.
- Using Composer to create business rules that let you customize statements for marketing purposes.

The following diagram shows a high-level view of the eStatement Manager application development process:



What is a DDF?

A Data Definition File, or DDF, contains the rules for finding and extracting data from an application's input data source. You create one or more DDFs using eStatement Manager's DefTool exclusively for use with your organization's particular data source format and application.

For each eStatement Manager application, it is recommended that you create one DDF expressly for use in indexing your input file, and additional DDFs for extracting different views of data for dynamic display on the Web.

The Indexer production job prepares a data source and the entire eStatement Manager application for live statement retrieval. Indexing adds statement summary data to the application database table for each data field in the DDF you publish expressly for the Indexer job. In this DDF, define only those fields you want indexed.

CAUTION: Be sure to include all fields you need to index when you create the indexing DDF. Although you can add new fields to a DDF later, you must manually add the new columns to your database index table.

Create additional DDFs with only the particular data you need to extract for dynamic display on the Web. eStatement Manager uses these DDFs on a regular basis during statement live retrieval to extract different presentations of account data as requested by Web users.

CAUTION: DefTool creates the DDF in XML format. Do NOT edit the XML directly, or the resulting DDF may be unusable. If you receive the message "Failed during internal validation" in DefTool, the schema validation has failed, and the DDF may have been altered. See *DDF*_error, where *DDF* is the name of your application, in the folder where your DDF is located, for details.

What is an ALF?

The Application Logic File, or ALF, contains the rules to present the data extracted from the original data source in a customized HTML format. You create one or more ALFs using the eStatement Manager Composer tool. The ALF (which is in XML format) is then used along with the DDF during live statement retrieval to display extracted account data on the Web, in email notifications, etc. The ALF also contains business logic (conditional statements that consider current statement data) for marketing and other customization purposes.

In addition to dynamic statement presentment, ALFs are also developed and used for the composition of static output, most commonly notification emails. An application can have multiple ALFs for use in presenting different statement Views.

The ALF defines:

- Formatting rules for presenting the data on an HTML template
- Paging, sorting, and filtering configuration information
- Business logic to customize and personalize a statement
- File names of the dynamic DDF and any HTML templates used for the statement view

eStatement Manager-specific tags are added to the HTML templates during the Composition process, indicating where the data should be inserted and where logic arguments are applied. These logic arguments, as well as other assembly instructions, are stored in the ALF.

You must create an ALF for each view, including a separate ALF for the notification email. You can also specify the conditions for the use of different ALFs.

The HTML Template

You use an HTML editor to create the HTM or HTML templates for presenting your statements online. However, they inherit some information about the fields, tables, groups and charts that are to be extracted and displayed when you map your data and other design elements in Composer.

TIP: Use placeholders in your template design. There are so many values in most bills and statements that it is easy to overlook one without a visual cue. This is particularly helpful if different people design the templates and map the values.

- You can use Composer as an HTML editor while developing your application's ALFs for presentment.

What is DefTool?

DefTool is the eStatement Manager GUI application design tool you use to define the rules for extracting data from your application's data input file. DefTool presents the source file in a standard format, making it easy to see the statements and identify the data you want to extract. You can visually select the data elements required for presentment and business logic.

You must create a rule for extracting each data field, table, and group that you want to display in an online statement. As you work in DefTool, you can use the Extraction Simulator to test each data extraction definition.

The result of using DefTool is a Data Definition File (DDF) containing the data extraction rules for your application.

What is Composer?

Composer is the eStatement Manager GUI design tool you use to map the data extracted from your data input file to an HTML template for online presentation.

The result of using Composer is an Application Logic File (ALF) for your application and an edited version of one or more HTML templates.

When a customer makes a request to view a statement in a live eStatement Manager application, the eStatement Manager server uses the information in the ALF to locate the associated DDF, HTML template, digital images, and the data input file for extracting and present the statement.

Using the National Wireless Sample Application

A sample eStatement Manager application named "NatlWireless" is included with your system to let you demonstrate and practice using the eStatement Manager design tools (DefTool and Composer) with sample files and data. NatlWireless simulates telecommunication statements in an ASCII file format.

With some procedures in this guide, specific instructions for using NatlWireless are included so you can follow along (if you choose). The National Wireless input instructions appear in parenthesis after each step.

The sample application files can be found in <EDX_HOME>\samples\NatlWireless.

The NatlWireless application consists of the following files:

NatlWireless File	Purpose
NatlWireless.txt	Sample data input file (source).
NatlWireless.htm	HTML template for presenting the summary statement view (named HtmlDetail).
NW_LocSummary.htm	HTML template for presenting the detail statement view (named NW_LocSummary).
NWEmailDefault.htm	Default HTML template for notification emails.
NWEmail.htm	Alternate layout for notification emails (its use is determined by business logic).
IndexerJob\ NatlWirelessIndexer.DDF	Defines fields for indexing the NatlWireless data.
NatlWireless.DDF	Contains the customized rules and business logic for extracting each field from each page of the original statement file for presenting different views of the data in online statements.
NatlWireless.ALF	Contains the customized rules and business logic for mapping and presenting the extracted data in an online summary statement.
NW_EmailDefault.ALF	Contains the customized rules and business logic for creating notification email.

After Creating an eStatement Manager Application

When you are finished using DefTool and Composer to create your application, refer to the *Administration Guide for Oracle Siebel eStatement Manager* for instructions on how to use the eStatement Manager Command Center to implement and run an application in a live production environment.

3 Overview of Creating a DDF with DefTool

Before Creating Your DDF

Before you can begin creating a DDF for your application, you must find a sample data source file that is representative of the format in which you deliver your statements. This file must contain a broad and varied sample of statement styles and data types.

Use this sample data input file to:

- Identify the data you want to extract
- Define an indexing DDF and view DDFs in DefTool

Much of the information required to identify data for extraction and create appropriate data extraction rules results from the Oracle Bill Mastering process. You can use this book to assist in the Bill Mastering process.

Steps for Creating a DDF

For each application, you must create one DDF to use exclusively with the Indexing job, and additional DDFs as needed for extracting data in various live presentment views.

For the indexing DDF, define all the fields you need to index for the application.

CAUTION: Be sure to include all fields you need to index when you create the indexing DDF. Although you can add new fields to a DDF later, you must manually add the new columns to your database index table.

For view DDFs, define the fields, tables, groups, and other rules necessary to extract the particular view of data you want to present. Your application may require multiple view DDFs.

To create a DDF for your application:

1. **Run DefTool.** From the Windows Start menu, select **`Programs>eStatement Manager>DefTool`**.
2. **Create a new (empty) DDF**. Follow the procedure for the particular type of data input file your application will use (DJDE, AFP, DB Extract, line printer, etc.). You must define your DDF using a sample data source file that is representative of the format in which you deliver your statements. The file you use to define the DDF must contain a broad and varied sample of statement styles and data types.
3. **Define the document style.** The document style defines how eStatement Manager can identify the start of a new statement in the data input file. It defines the first page of a new document based on a unique pattern that signals the start of a new document in the data source.
4. **Define the primary key**. The primary key contains the rules for locating the field defined as the unique identifier for each statement (and therefore recipient) in a billing cycle.

5. **Define additional page styles (if necessary)**. If pages subsequent to the document style (first page) contain different content, you want to define additional page styles for each different type of page in your input file statements.
6. **Define the necessary field, table, and group extraction rules.** You must define rules for extracting each element from your data input file that you want to display online. The rules tell eStatement Manager where to find the data (general location) in the file, and the type of information it contains (content). As you work in DefTool, use the Extraction Simulator to test each data extraction definition. DefTool also lets you apply formatting specifications to field and table column values after they are extracted (post-conversions).
7. **Save the DDF.**

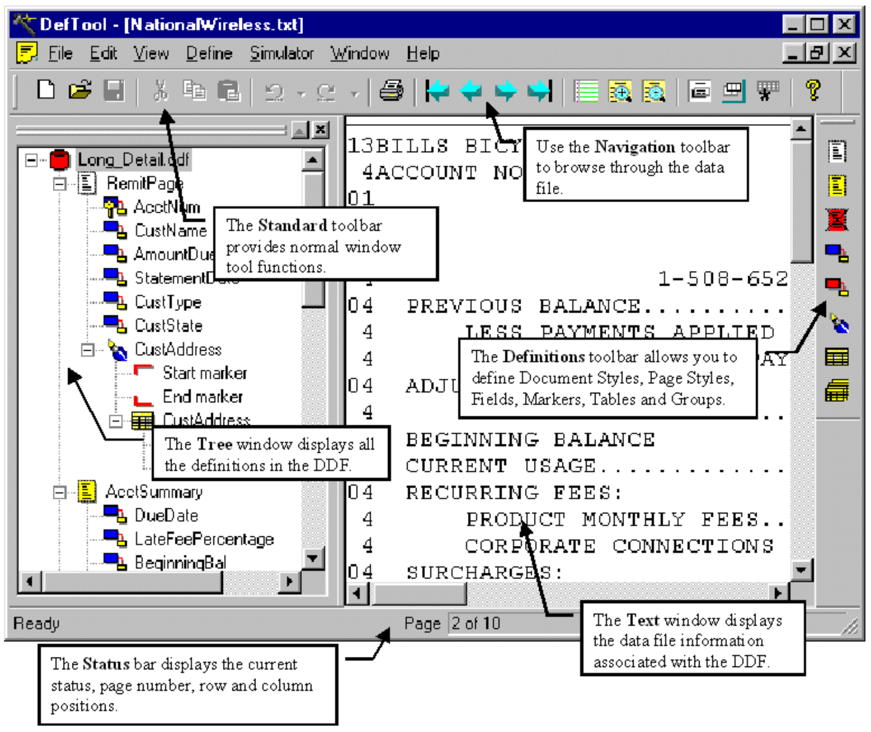
Your DDF is complete when it contains all the rules necessary to extract statement data from your input file for display online. When your DDF is complete, you can proceed to use Composer to map the fields to an HTML template for display and define any business logic you want to use.

The DefTool Interface

The DefTool interface consists of:

- The DDF tree on the left side
- The DefTool work area (search area) on the right side
- Menus
- Toolbars

The following diagram shows an open DDF in DefTool. The tree on the left shows the elements defined for extraction from the data input file. The work area on the right displays the content (statements) of the associated data input file.



```
13BILLS BICY
 4ACCOUNT NO
01
                        1-508-652
04  PREVIOUS BALANCE.........
 4      LESS PAYMENTS APPLIED
 4                           AY
04  ADJU
 4
    BEGINNING BALANCE
    CURRENT USAGE............
04  RECURRING FEES:
 4      PRODUCT MONTHLY FEES..
 4      CORPORATE CONNECTIONS
04  SURCHARGES:
```

The **Standard** toolbar provides normal window tool functions.

Use the **Navigation** toolbar to browse through the data file.

The **Definitions** toolbar allows you to define Document Styles, Page Styles, Fields, Markers, Tables and Groups.

The **Tree** window displays all the definitions in the DDF.

The **Status** bar displays the current status, page number, row and column positions.

The **Text** window displays the data file information associated with the DDF.

The DDF Tree

- The left side of the DefTool window shows all the rules created for the data input file to identify the various statement elements, including page styles, fields, tables, etc, in a tree format.

To expand a list of elements:

- Click on the tree pages, tables, etc.,

(When you first create a new DDF there are no elements in the tree.)

The DefTool Work Area (Search Area)

- The DefTool work area shows a digitized rendering of the sample data source in page-view format. This enables you to identify the search area location of elements for extraction graphically.

You can zoom in to get a close-up view of the file or zoom out to see more of the data file at a reduced size.

To enhance readability in DefTool, you can optionally view the search area, in colored bar-paper style shading.

To zoom in on a file:

- Select **`View>Zoom In`** or click .

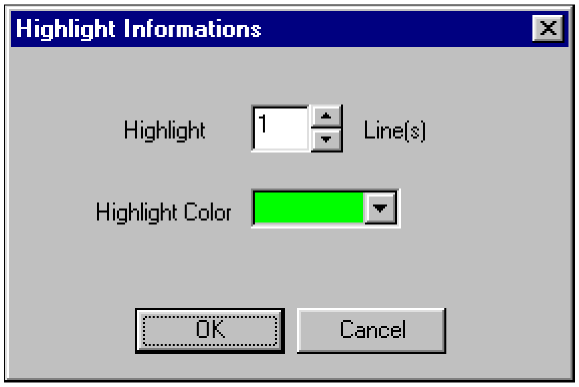
To zoom out of a file:

- Select **`View>Zoom Out`** or click .

TIP: To get a high-level view of a statement page, select **`Zoom Out`** several times until the entire page is visible in the work area. This is a good way to determine if the data is properly rendered on the page.

To view the data input file with color bar shading:

1. Select **`View>Color Shading`** or click . The Highlight Information dialog box appears.



Highlight 1 Line(s)

Highlight Color 🟩

2. Specify the number of lines to highlight.
3. Select a color from the Highlight Color drop-down menu.
4. To view more colors, select **`More`**. The Color dialog box appears.
5. Select a color.
6. To define a custom color, specify the color using the Color Matrix, the hue/saturation/luminosity, or the red/green/blue definitions. Click **`Add To Custom Colors`**.
7. Click **`OK`** at the next two dialogs.

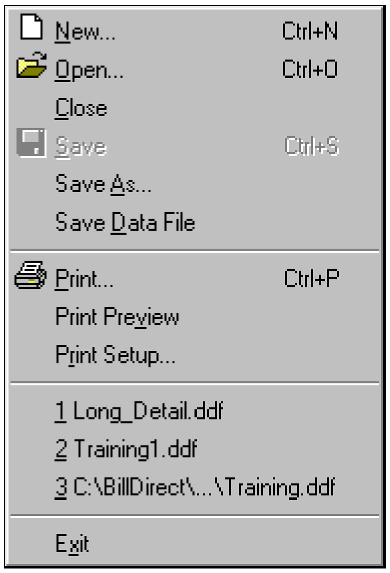
TIP: To remove the highlight bars, click the Highlight icon and select white as the Highlight Color.

Menus

The Standard Toolbar contains seven selections—each containing a number of user definable options.

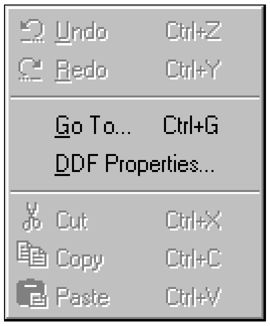
File Menu

DefTool offers all the standard File Menu options, such as New, Open, Close, Save, Save As, Save Data File As, Print, Print Preview, Print Setup, and Exit. The Print Preview option enables a preview of the entire page displayed within the Work Area. This is very useful for quickly viewing the entire page to see where data falls within the defined boundaries.

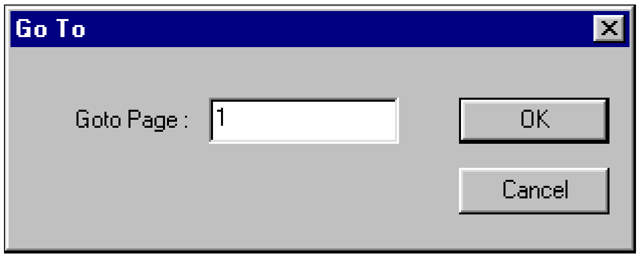


Edit Menu

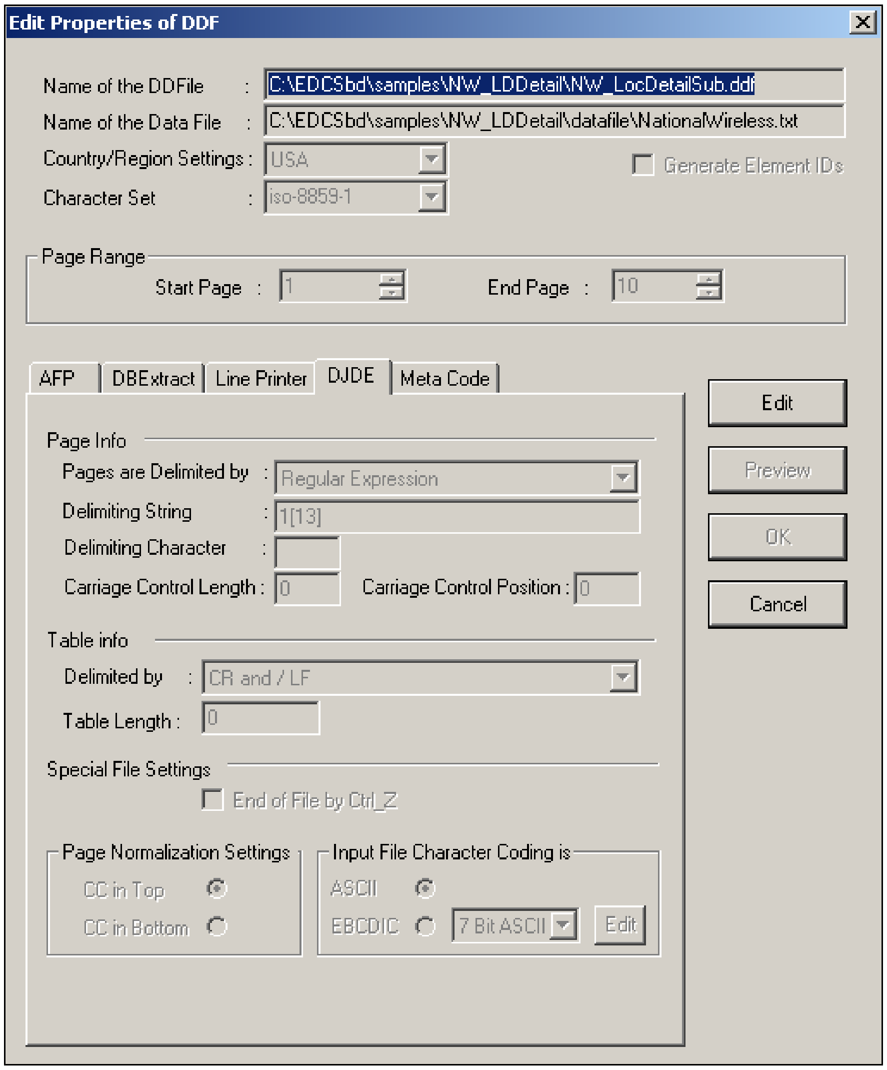
The Edit menu allows moving quickly to any page with the **`Go To`** option. DDF Properties lets you change the data file properties of a DDF. The most recent operation can be undone (Undo), or an undone action can be redone (Redo).



For example, you can go to a specific page within a file:



The **`DDF Properties`** option opens a window (shown below) that displays all the settings entered during the creation of the DDF, such as the Page Range, Normalization Characteristics, and Special AFP Settings. You can edit many of these properties by clicking on the **`Edit`** button, allowing any necessary changes without re-creating the settings in an entirely new DDF.



Edit Properties of DDF

Name of the DDFile : C:\EDCSbd\samples\NW_LDDetail\NW_LocDetailSub.ddf
Name of the Data File : C:\EDCSbd\samples\NW_LDDetail\datafile\NationalWireless.txt
Country/Region Settings : USA
Character Set : iso-8859-1
Generate Element IDs

Page Range
Start Page : 1
End Page : 10

AFP | DBExtract | Line Printer | DJDE | Meta Code

Page Info
Pages are Delimited by : Regular Expression
Delimiting String : 1[13]
Delimiting Character :
Carriage Control Length : 0
Carriage Control Position : 0

Table info
Delimited by : CR and / LF
Table Length : 0

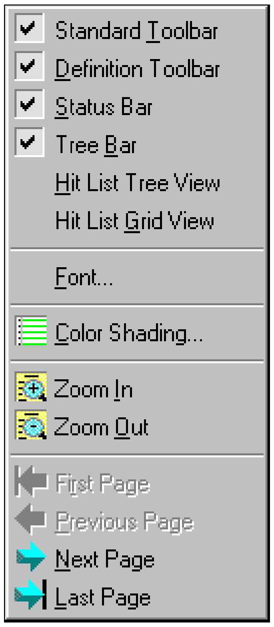
Special File Settings
End of File by Ctrl_Z

Page Normalization Settings
[✓ CC in Top] CC in Bottom

Input File Character Coding is
[✓ ASCII] EBCDIC 7 Bit ASCII

View Menu

The **View** menu option allows customization of the view for the DefTool main console and the work area.



You can elect to display any combination of the following items in the DefTool:

Standard Toolbar — Located at the top of the DefTool main console and contains standard Windows tools, such as New, Open and Save.

Definition Toolbar — Located on the right border of the DefTool main console and contains the tools needed to create rules for statement and page style detection and data extraction

Status Bar — Located in the bottom frame of the DefTool main console and contains information about the current page in the work area and the most recent row/column coordinates in the work area where the mouse pointer resides.

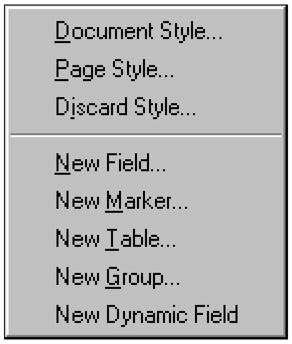
Tree Bar — The left pane of the DefTool home window, containing the tree representation of the DDF rules.

The **Font…** Option specifies a font for displaying the data source text in the Work Area.

Color Shading defines horizontal color bars that will overlay the text in the work area. A specific number of rows can be highlighted.

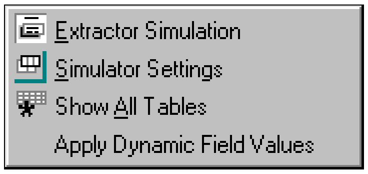
Define Menu

The Define options can be used to specify document styles, page styles, fields, markers, and tables. The Define options are also available in their own toolbar.



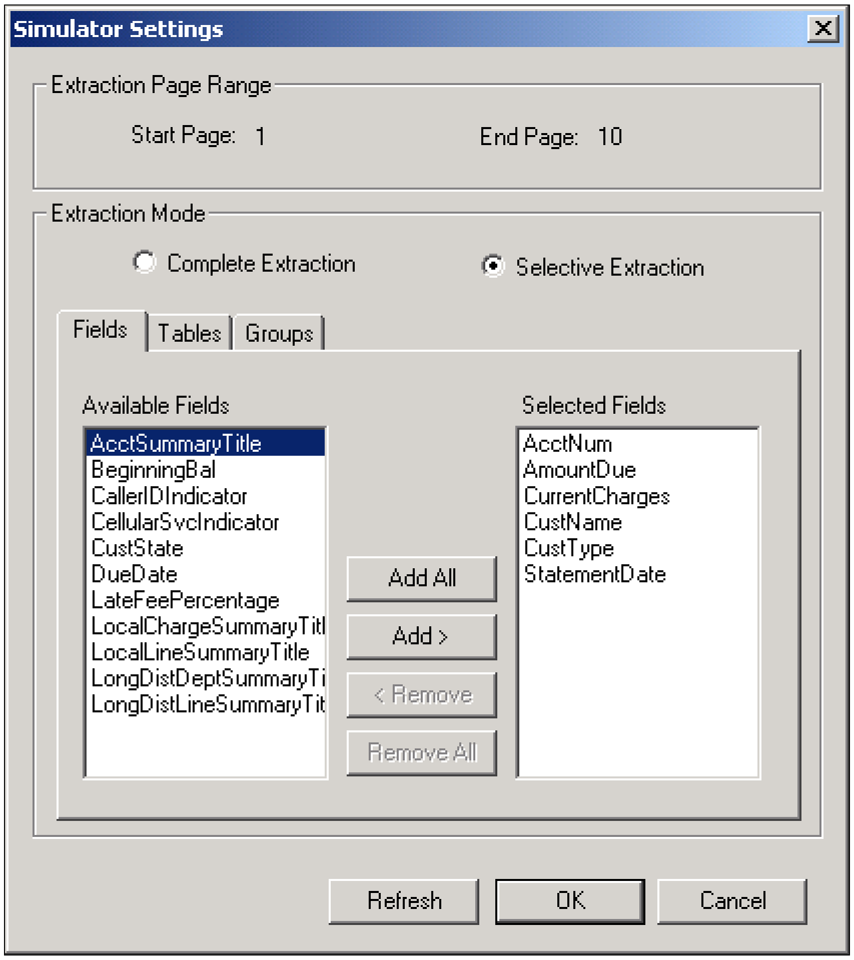
Simulation Menu

This brings up the Extractor Simulator used to test the definitions.



To test the extraction rules against the data source Page Range, select the **`Extraction Simulation`** option. By default, the Extraction Simulator will simulate extraction of all the rules.

To perform selective extraction, i.e. to select particular rules for testing extractions, choose the **`Simulator Settings`** option (as shown in the following figure).



Start Page: 1

End Page: 10

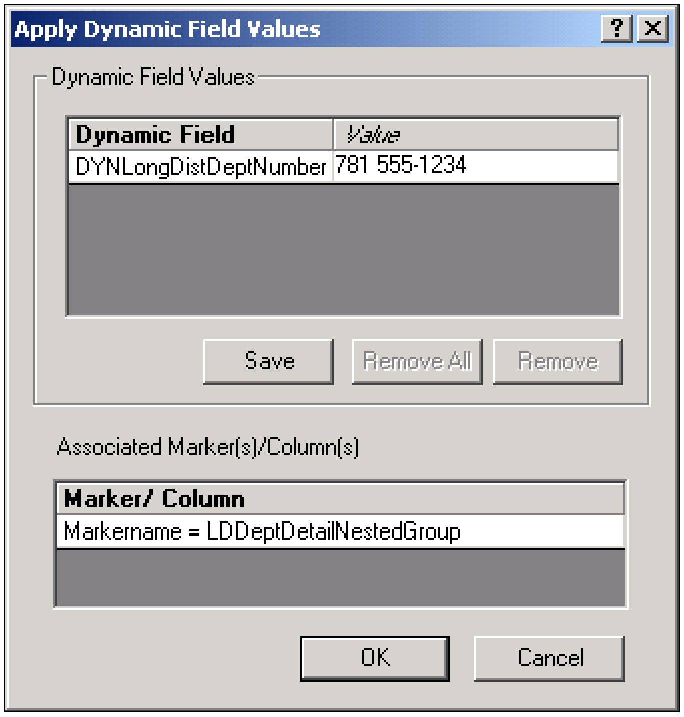
[Complete Extraction] [✓ Selective Extraction]

Available Fields	Selected Fields
AcctSummaryTitle	AcctNum
BeginningBal	AmountDue
CallerIDIndicator	CurrentCharges
CellularSvcIndicator	CustName
CustState	CustType
DueDate	StatementDate
LateFeePercentage	
LocalChargeSummaryTitl	
LocalLineSummaryTitle	
LongDistDeptSummaryTi	
LongDistLineSummaryTit	

The selections made here will be saved to the Registry of the development server, so that each simulation will provide the results for the data items you want to view.

By default, the DefTool Simulator will show the first occurrence of a table within a group, and allow navigation to the subsequent occurrences as desired. To view all occurrences of your group's tables, select the **`Show All Tables`** option. This option works in a toggle mode; you do not need to select it repeatedly before each extraction.

If you are building an extraction rule to retrieve a specific group occurrence by dynamic pattern matching, you can test the extraction by selecting the **`Apply Dynamic Field Values`** menu option.



Dynamic Field	Value
DYNLongDistDeptNumber	781 555-1234

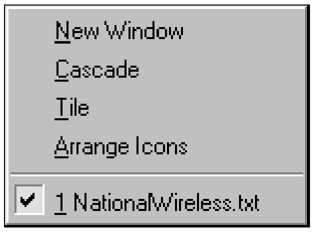
Marker/ Column
Markername = LDDeptDetailNestedGroup

From the dialog that appears, specify a possible occurrence of the dynamic field by typing it into the Value field. You may elect to save this string to the Registry to streamline your testing process.

TIP: The use of the Apply Dynamic Field Values option requires that dynamic pattern matching be used in at least one of the group markers, table markers, or table columns.

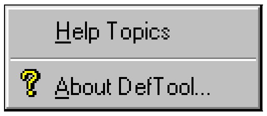
Window Menu

The Window options provide the standard tile, cascade, and arrange capabilities.



Help Menu

Provides access to the online Help for DefTool and Composer and to the About box.



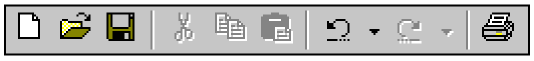
Toolbars and Icons

There are two primary toolbars in DefTool:

- Standard Toolbar
- Definition Toolbar

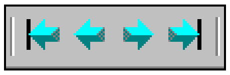
Standard Toolbar

The Standard Toolbar, shown below, contains icons that allow quick access to the basic File, Edit, and View functions. By simply holding the mouse over the icon, a "Quick Help" description will appear.



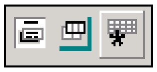
Navigation Icons

The Navigation icons (First Page, Previous Page, Next Page and Last Page) allow movement from page to page of the sample data source within a previously selected page range.



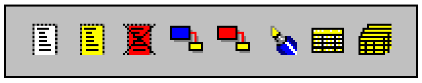
Simulator Icons

The simulation icons are used to test the data extraction rules that you create in the DDF.



Definition Toolbar

The Definition Toolbar is used to activate the various data extraction options. These options are: Document Style, Page Style, Discard Style, Field, Dynamic Field, Marker, Table, and group. With the exception of the Table and Group icons, a tool selection will remain until another tool is selected.



TIP: You can drag and drop the Definition Toolbar next to the Standard Toolbar.

Document Style

The Document Style icon is used to create the Start of Statement setting. This mandatory setting is placed in all DDFs to determine where the statements begin. In addition to delimiting statements in a data source, this setting is also essential during the data source indexing process, when it indicates where the statements of the data source begin.

A document style represents the entire document for a given customer. The DocStyle identifier (data) must be a unique element that appears at the beginning of each document.

This data is stored in the index for all statements, and is used to locate statements in a data source during dynamic data retrieval.

CAUTION: The New Document Style icon is always the first tool that is used in your DDF. You must create the document style before creating any other extraction rule. The document style is only used once in each DDF.

Page Style

The Page Style icon defines rules for detecting the "look and feel" of a unique style of page. The information needed to determine how many additional page styles your statements may have, if any, would be developed during your Oracle Statement Mastering sessions.

In summary, a unique page style would not only have a different visual layout from the other pages in a statement, but it would also contain data elements that would not appear on other page styles. More than one statement page may use the same page style. An example of different page styles can be found on a statement from a telecom company. The first page of each statement is often a remittance page, presenting customer name, account number, and total amount due. The subsequent pages of the statement may have a different layout and contain different data elements. For example, one or more detail pages may present itemized tables for different calling plans. A new page style would be created and rules to extract the call itemization data would be associated with the detail page style, allowing the extraction components to only consider pages of this style when data elements are dynamically extracted.

Discard Style

You would define a discard page style in the same fashion as an additional page style, however, the extraction components detect any pages that fall under this rule and completely disregard them during data extraction. The discard page style defines the page styles that are NOT considered for any type of data extraction.

New Field

The New Field tool defines the rules for extracting data items whose value exists on one row and appears once per statement (multiple occurrences of the same value may appear on a statement, such as an Account Number, but there is only one real value for the Account Number in each statement).

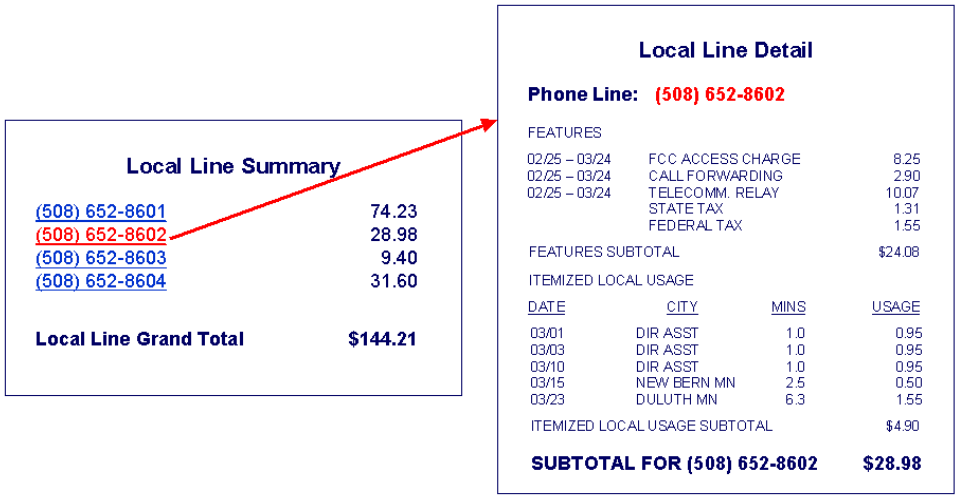
Sometimes a field location may vary widely across a statement page. For example, the Total Amount Due may appear close to the top of a statement with few details, or near the bottom of a longer bill. In this case, you will need to create multiple locations for the field and an additional extraction rule to provide extra assurance that the correct data will be extracted. This additional extraction rule is called a field anchor and is also created with the New Field tool.

TIP: Use anchors with Markers and page styles also. Create them in the same way as fields — using the appropriate tool and selecting **Anchor For** in the dialog box.

New Dynamic Field

The New Dynamic Field tool creates a dynamic field that is used in the pattern of a group marker, table marker, or a table column. You may want to quickly provide your end-users with a hierarchical and organized view of their statement data, especially if the statements are very large. For example, you may want to present the Summary information of a telecom bill and allow your recipients to "drill down" into a more detailed statement view by providing a link that will retrieve the Detail data when selected. If the statement is very large, it may be useful to design the links in the Summary view to perform retrieval of only a section of the Detail data.

For example, to retrieve Detail data for the Phone Line 508-652-8999 by clicking on a link in the Summary view, you can order that only detail that pertains to 508-652-8602 is extracted and presented. Part of the configuration to provide for this functionality starts at the DDF level. You can use a dynamic field in the pattern of a marker of the group that extracts phone line detail. During live retrieval, the pattern is dynamically filled with a specific phone line, such as 508-652-8602, guaranteeing that the extracted data pertains only to this Phone Line.



Local Line Summary

(508) 652-8601	74.23
(508) 652-8602	28.98
(508) 652-8603	9.40
(508) 652-8604	31.60
Local Line Grand Total	**$144.21**

Local Line Detail

Phone Line: **(508) 652-8602**

FEATURES

02/25 – 03/24	FCC ACCESS CHARGE	8.25
02/25 – 03/24	CALL FORWARDING	2.90
02/25 – 03/24	TELECOMM. RELAY	10.07
	STATE TAX	1.31
	FEDERAL TAX	1.55

FEATURES SUBTOTAL $24.08

ITEMIZED LOCAL USAGE

DATE	CITY	MINS	USAGE
03/01	DIR ASST	1.0	0.95
03/03	DIR ASST	1.0	0.95
03/10	DIR ASST	1.0	0.95
03/15	NEW BERN MN	2.5	0.50
03/23	DULUTH MN	6.3	1.55

ITEMIZED LOCAL USAGE SUBTOTAL $4.90

SUBTOTAL FOR (508) 652-8602 **$28.98**

New Marker

The New Marker tool is used to define the start and end markers of tables and groups. Select a data string (static or variable) that always appears at or near the start of a table or group, and use the New Marker tool to create the rules to find that data string. The same tool is used to define rules to locate the data string that signifies the end of a table or group.

New Table

The New Table tool creates rules for extracting one or more related data strings. These data strings may have multiple occurrences, and are extracted as table columns. A table can have only one column, such as a customer address section. The customer address is a field that can occur on more than one line. Since this data does not qualify as a field (more than one row of data), it must be extracted as a table.

A good example of multiple columns that are related to each other in some way are the components of an itemized phone call, containing a date, time, location, phone line called, number of minutes and charge. These data are not appropriate fields because they have iterations within a statement. A call itemization list can contain thousands of items, with thousands of occurrences in each of the columns suggested above. The table functionality is useful because it does not require knowledge of the number of rows a field may have, or how many occurrences of call items appear. The extraction components use the location of the start and end marker to determine how many rows should be searched for data.

Before creating a table, the start and end markers must be selected and defined using the New Marker tool. Then use the New Table tool to define the table name, select the markers used, and define the columns.

New Group

The New Group tool is used to define which tables or groups should be considered for extraction together. A group can be a collection of tables, (or other groups), that appear in statements as a pattern. The pattern may have several iterations in a statement. The start and end location of each group occurrence must be determined and a start and end marker defined respectively for the data string. Next, identify and define the items that will be part of the group. These items can be tables or other groups. (Fields cannot be selected as part of a group.) Use the New Group tool to define the group name, select the markers to be used and select the items (tables and groups) that are in the group.

Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length.

Cutting, Copying, and Pasting

You can use cut, copy, and paste functionality to move or copy the following DDF content within a DDF or to another DDF:

- Page styles
- Fields, including group fields
- Markers
- Tables and columns
- Groups

Using cut, copy, and paste functions saves time and reduces errors when building multiple application views.

(Note that you can only open a second DDF with a second data input file.)

If you copy and paste an entity into the same location as the original, you may be prompted to give it a new name. A "location" just refers to another place within the same DDF.

When you cut and paste an item, DefTool also moves or copies any associated child items (entities that belong to them) as child items in the tree.

The basic rules and behavior associated with copying each type of DDF entity are described here.

Page styles

When you copy and paste a page style (in the same DDF), DefTool prompts you to enter a new name. DefTool copies all the entities belonging to that page style onto the new one. Therefore, if a marker has a location in one page style, DefTool creates another marker in the target page style as well. (DefTool displays the marker under the root of the tree, not the node of the page style.)

Fields, group fields

You can cut, copy, and paste fields within the same page style or to a different one. You can also cut, copy, and paste group fields from one group to another.

If you paste a field within the same page style, you must provide a new name for the field.

If you paste a field to another page style, DefTool adds it as another location, so you do not need to rename the field.

Markers

You can move or copy a marker pair (start and end markers) to the same or another page style. DefTool moves or copies the associated tables and groups with the marker pair.

If you copy a marker pair to another page style, DefTool adds an additional location to the markers and displays the marker in the Definition Tree root (and not under the page styles).

If the source and target page styles are the same, DefTool adds a new location to each marker, and markers remain under their page node.

You can also move or copy *individual* start and end markers to the same or to another start or end marker.

Note that if you rename a marker location, it is equivalent to renaming the marker.

Tables, columns

To copy a table, you must paste it to the marker pair you want to associate it with.

When you copy and paste a table, DefTool prompts you for a new name and uses this name as a prefix to the columns of the new table. DefTool displays the renamed table under the marker node in the Definition Tree.

When you paste a table, DefTool copies or moves the associated anchor and columns.

You can also cut or copy a column from one table to another table. DefTool prompts you for a new name. If you cut a column that is an internal anchor, the anchor becomes external.

Groups

You can copy and paste a group to the same set of markers or to a new pair you want to associate the group with. When you copy and paste a group you must provide a new group name (only the top-most group is a copy; the table and column names remain the same).

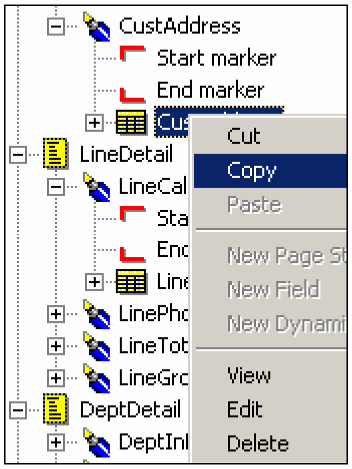
When you paste a group, DefTool moves all child groups and tables belonging to the group.

You can also copy a group and paste it as a child group under another parent group (in this case the group name stays the same).

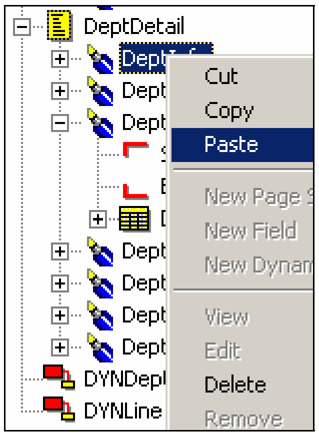
To move or copy DDF entities:

- With the DDF open in DefTool, right-click the item in the tree you want to cut or copy and paste. To select multiple items, press **`Ctrl`**, then right-click on each item. To select a range, select the first item, then press **`Shift`** and click the last item.

- Choose **`Cut`** or **`Copy`** from the right-click menu.



- Right-click the target location or item in the open DDF (or switch to another open DDF and right-click the target item there) and select **`Paste`** from the right-click menu.



- If DefTool prompts you for a new name, enter a unique name for the item and click OK.

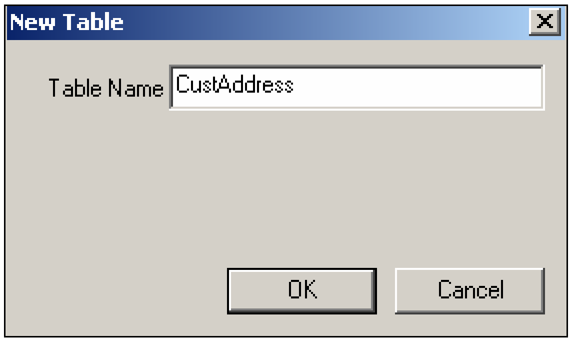
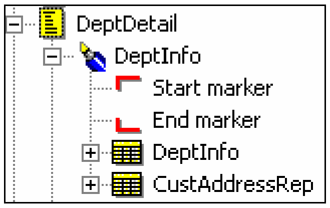
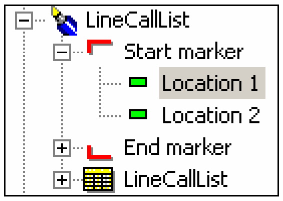


Table Name: CustAddress

- DefTool redisplays the Definition Tree to reflect the result of your edits.



- Note that if you copy and paste a marker pair to a different page style, DefTool displays the markers under the root, with multiple location nodes beneath it. To see which page style a location node represents, right-click on the node, such as "Location 1" and select **`Edit`** from the right-click menu. The Marker Properties dialog shows the page style name.



- Click 💾 to save the DDF.

4 Creating a New Data Definition File

Overview

The DDF is used during live production to instruct the parser to extract key data elements from the defined input file type. eStatement Manager can accept input data from various file types, including:

- AFP (Mixed Mode and Fully Composed)
- DB Extract
- Line Printer
- DJDE (Dynamic Job Descriptor Entry)
- Metacode

Essential to the creation of a DDF is the identification of the data source and its properties. When creating a new DDF based on a sample data source, you submit specific information about the data source in order for the DefTool to convert settings intended for a print- or storage-style format, into settings to render the data source in digital format. The information required by DefTool varies depending on the data source format and unique settings. The Oracle Application Profile Document outlines the questions to ask about the data file. These answers, when used with the DefTool Configuration Wizard, will assist you in creating a DDF.

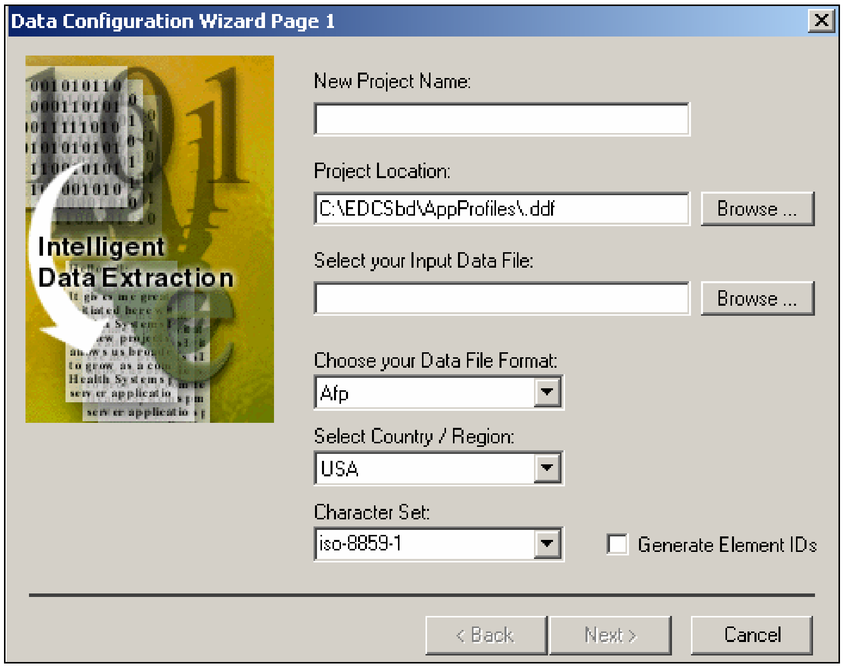
The Data Configuration Wizard pages for creating a DDF will vary, based on the file type selected and are presented by input data file formatting, starting with the DJDE format.

Creating a DDF for DJDE Files

The NatlWireless sample data source (NatlWireless.txt) is an ASCII text file from a Dynamic Job Descriptor Entry (DJDE) formatted input data file.

To create a new DDF for a DJDE data input file:

1. Select **`Start>Programs>eStatement>DefTool`**. The DefTool window appears.
2. Select **`File>New`** from the Main menu. Data Configuration Wizard Page 1 appears.



New Project Name:

Project Location: C:\EDCSbd\AppProfiles\.ddf

Select your Input Data File:

Choose your Data File Format: Afp

Select Country / Region: USA

Character Set: iso-8859-1

☐ Generate Element IDs

3 In the **New Project Name** field, enter the project name. This can be the application name (DDN) or the name of the statement view. (Enter **NatlWireless**.)

CAUTION: The project name is the descriptive application name assigned during project planning. Select a meaningful name. This name will be associated with the constructed DDF. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length.

4 Next to the Project Location field, select **Browse** and navigate to *<EDX_HOME> \Samples* and enter **ProjectName.ddf**. (Enter NatlWireless\NatlWireless.ddf.)

CAUTION: The name of the DDF will be affiliated with the project name entered above, and should therefore match.

5 Click **Open**. The Data Configuration Wizard Page 1 returns.

6 To select your input data file, click the **Browse** button to the right of the field and navigate to the location of the data source that will act as the broad sample of statements for building the rules for data extraction. (Select *<EDX_HOME>\Samples\NatlWireless\NatlWireless.txt*.)

7 Click **Open**. Data Configuration Wizard Page 1 appears.

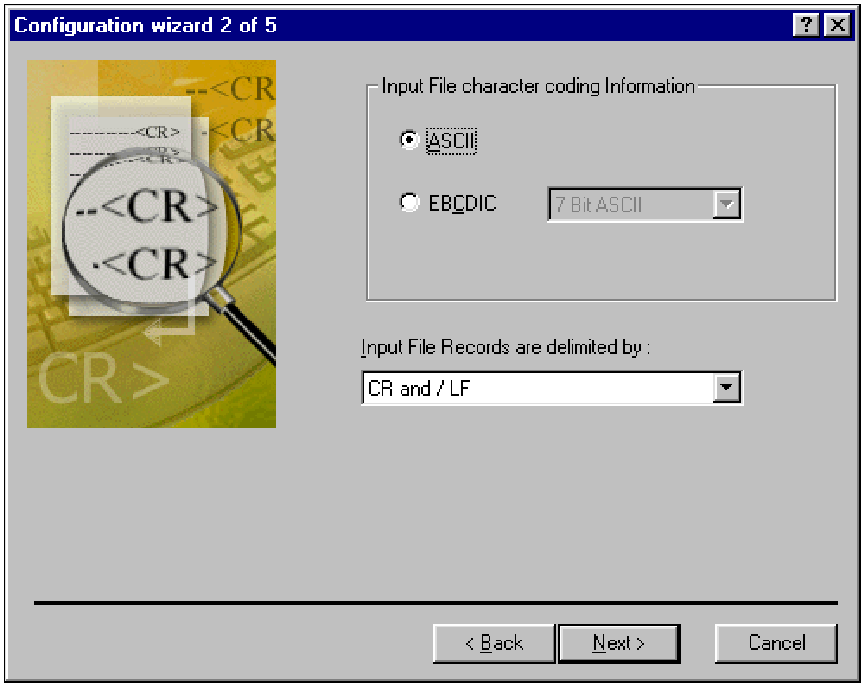
8 To choose your data file format, click the corresponding dropdown list and select the data format. This will be the expected data format of all data sources for extraction. (Select **DJDE**.)

9 Select the country/region from which the data source originates.

10 Select or type the name of a character set used in the input file. DefTool cannot verify a typed set name; type carefully and be sure to enter the name of a valid character set.

11 To enable the system to generate Element IDs for unique line item identification, click the Generate Element IDs box. See page 163 for more information about using Element IDs.

12 Click **Next**. Configuration Wizard 2 of 5 appears.


Configuration wizard 2 of 5

Input File character coding Information

ASCII [✓]

EBCDIC — 7 Bit ASCII

Input File Records are delimited by : CR and / LF

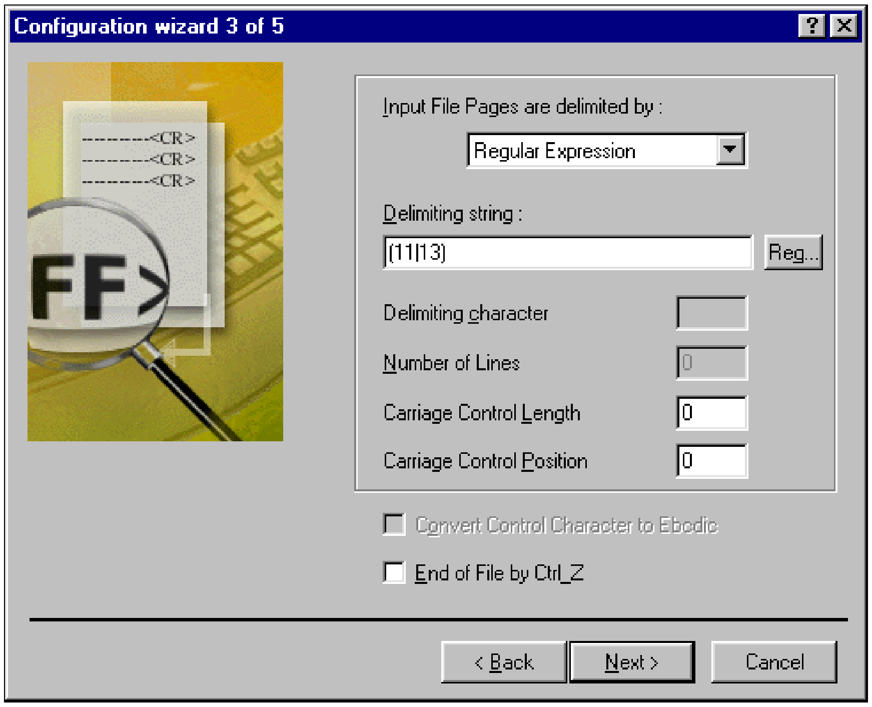
Depending on the data format, the DefTool can auto-detect configurations. For example, the DefTool knows that AFP files store data in the EBCDIC, rather than the ASCII, character set and will use a Carriage Return <CR>as the separator of file records.

Text, database extraction or line printer data sources can vary widely in format and design, and frequently do not contain the information needed to set the layout of the statements. Thus, the DefTool cannot automatically obtain settings, such as Character Set or table Separators. A general rule of thumb is that the less structured the data source, the more information the Data Configuration Wizard will require.

13 Select the appropriate type of input file character coding information. This informs the DefTool about the character set that is used in the input data source, and the mode of conversion to ASCII, if necessary. (Select **ASCII**.)

14 Select the method that input file records are delimited by, from the corresponding dropdown list. (Select **CR/LF** (Carriage Return and Line Feed).)

15 Click **Next**. Configuration Wizard 3 of 5 appears.



Input File Pages are delimited by : Regular Expression

Delimiting string : (11|13)

Reg...

Delimiting character

Number of Lines 0

Carriage Control Length 0

Carriage Control Position 0

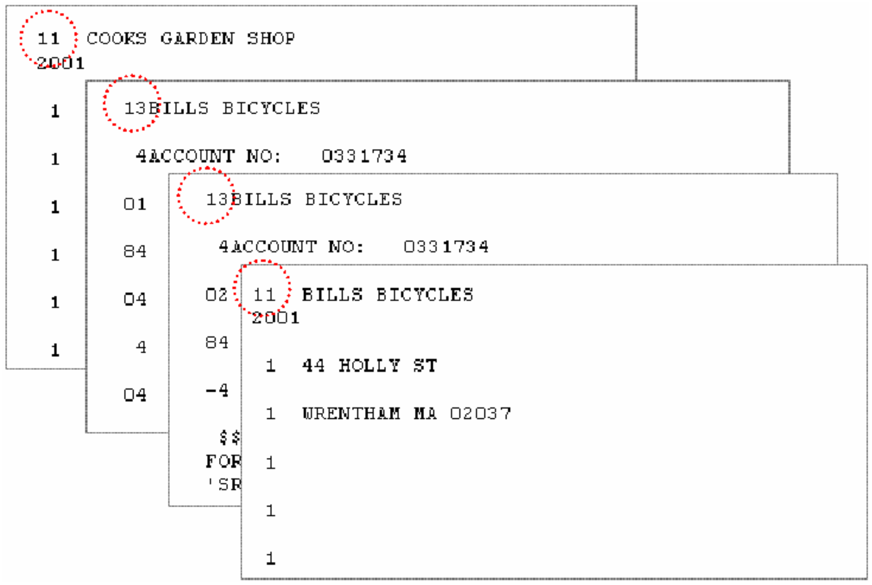
☐ Convert Control Character to Ebcdic

☐ End of File by Ctrl_Z

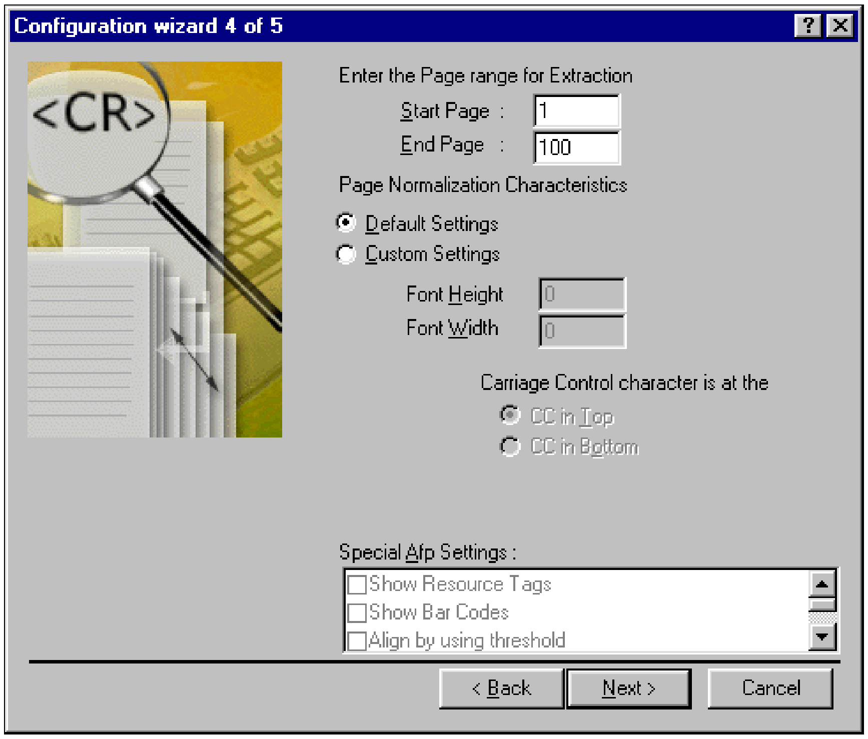
< Back | Next > | Cancel

16 Specify the method that input file pages are delimited by from the corresponding dropdown list. (This information is usually obtained from your Oracle Statements Mastering sessions.)

From the statement page segments of the National Wireless data shown below, it is determined that all of the Input File Pages contain the string 11 (on first pages) or a 13 (on subsequent pages), starting at the top left corner of the page. Therefore, the **`Input File Pages`** are delimited by **`Regular Expression`**, and the **`Delimiting String`** (using eStatement Manager regular expression syntax) is **`(11|13)`**. (Use this expression for the National Wireless input file page separator.)


11 COOKS GARDEN SHOP
2001
1
1
1
1
1
1
13BILLS BICYCLES
4ACCOUNT NO: 0331734
01
84
04
4
04
13BILLS BICYCLES
4ACCOUNT NO: 0331734
02
84
-4
$$
FOR
'SR
11 BILLS BICYCLES
2001
1 44 HOLLY ST
1 WRENTHAM MA 02037
1
1
1

17 If carriage controls have a known length or position, enter those values in the corresponding fields of Configuration Wizard 3. (For National Wireless, accept the defaults of Carriage Control Length and Position = 0, and click Next.) Configuration Wizard 4 of 5 appears.



Enter the Page range for Extraction

Start Page : 1

End Page : 100

Page Normalization Characteristics

[✓ Default Settings] Custom Settings

Font Height 0

Font Width 0

Carriage Control character is at the

[✓ CC in Top] CC in Bottom

18 Enter the page range for extraction; i.e. Start Page = 1 and End Page = 1000. The Page Range for Extraction setting allows a temporary selection of pages for use in developing the extraction rules (which will eventually apply to all the pages in the data source). Again, this setting is temporary to the current DefTool session, and has no bearing on the number of pages that will be extracted from a data source with this DDF during deployment of the application. Also, if it is discovered during a DefTool session that the selected Page Range does not contain an adequate sample of data, the Page Range is easily adjusted to open more pages of the data source while working in the DDF. (For NatlWireless.txt, set the **Page Range for Extraction** to a **Start Page** of 1 and an End Page of 100.*)*

Select a page range that includes a broad enough sample of statements, page styles and data elements, so the rules built into the DDF will capture all statement data. Sometimes, the sample data source you use can be very large, for example 100,000 or more pages. Testing your DDF rules on such a large data sample will create long run times for the extraction simulation. . A much smaller sample of pages will often suffice for obtaining a breadth of data styles. A recommended Page Range for DDF development is 10,000 or fewer pages

19 For page normalization characteristics, select **Default Settings**. This allows the DefTool to detect the high-resolution coordinate system of the data source, and translate this into the optimal lower-resolution coordinate system that will manage the digital rendering of the data source without sacrificing the data itself. The DefTool uses the resolution information (300 dpi, for example) and the page size (8 ½ × 11, for example), both of which are usually stored in the data source, and translates this information to build a system that is:

- Of much lower resolution that is appropriate for a digitized view of the data, and

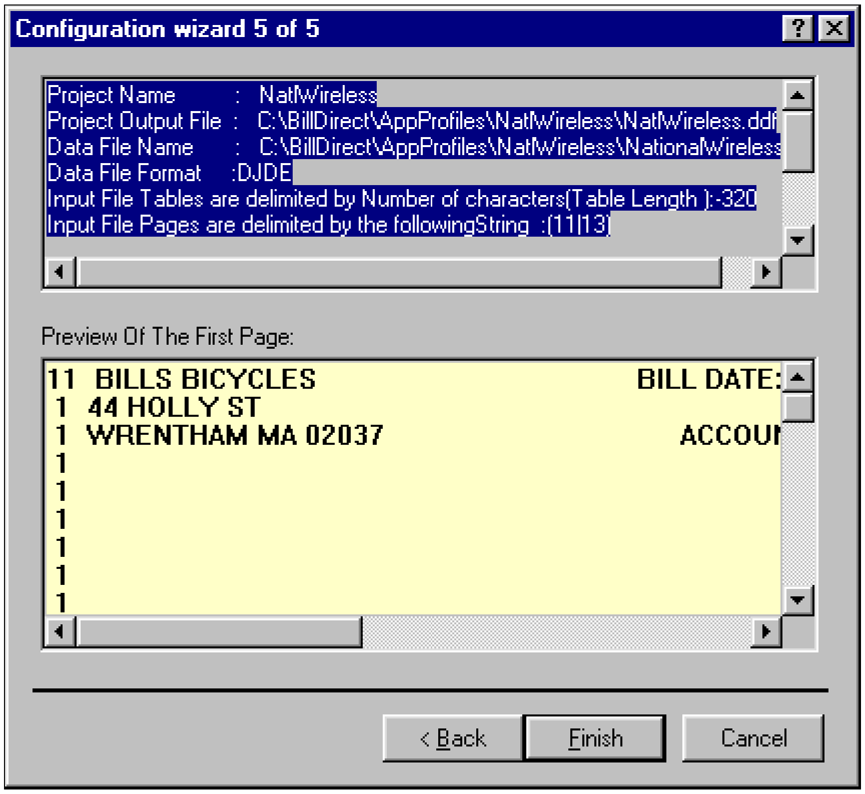
- High enough resolution to accommodate all data in the pages of the data source.

The DefTool is often able to determine the appropriate lower-resolution coordinate set by using the Default Settings. In some cases, however, a higher-resolution coordinate set is needed to render all page data properly and in its own byte locations. A good example of this is a formatted data source (such as an AFP) that contains data formatted to very small font character widths. If left to the Default Settings, the new coordinate system created in the DDF to render the data source may cause the smaller-font data to be overlapped by other data on the page. The DefTool will often need you to make Custom Settings, such as increasing the Font Width, in order to allow all data to "occupy their own bytes."

Font, height, and width relate to the grid size used when rendering print file data to the browser within DefTool, not the size of font displayed in the window. Adjusting the Font Height adjusts the number of vertical positions from the print file that map to a DefTool position. For example, Font Height 12 indicates that 12 print file positions map to one DefTool browser position. The same holds true for Font Width. Wider font spacing provides additional workspace on the page.

20 Enter the appropriate font height and width as required. (Since there are no very small font widths or other special formatting in *NatlWireless.txt*, so accept the **`Default Settings`** for the **`Page Normalization Characteristics`**.)

21 If your data source has any special AFP settings for extraction and presentation in the DefTool, check the appropriate items from the list in the corresponding scroll box. (For National Wireless, accept all other defaults, do not select any **`Special AFP Settings`**, and click **`Next`**.) Configuration Wizard 5 of 5 opens.



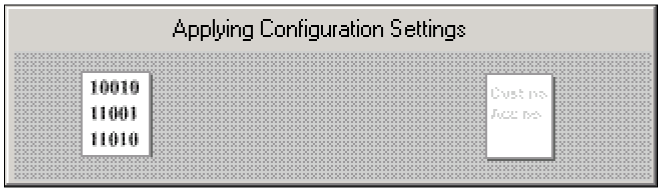
```
Project Name       :   NatlWireless
Project Output File :  C:\BillDirect\AppProfiles\NatlWireless\NatlWireless.ddf
Data File Name     :   C:\BillDirect\AppProfiles\NatlWireless\NationalWireless
Data File Format   :DJDE
Input File Tables are delimited by Number of characters(Table Length ):-320
Input File Pages are delimited by the followingString  :(11|13)
```

```
11  BILLS BICYCLES                          BILL DATE:
 1  44 HOLLY ST
 1  WRENTHAM MA 02037                          ACCOUN
 1
 1
 1
 1
 1
 1
```

The last page of every Configuration Wizard displays all of the settings created in the previous Configuration Wizard pages, regardless of the data format. It also provides a preview of the first page as it (and other pages) will be rendered in the DefTool work area. Scroll through this view to determine at first glance if the DDF properties are sufficient to accomodate all the data on the data source pages.

Confirm that the DDF properties match those presented in the configuration wizard and that all the data on the first page of Statement 1 appears in the preview of the first page.

22 Click **Finish**. The Applying Configuration Settings splash graphic appears, and the data source will appear in the DefTool work area. (A DDF icon appears in the Application Tree with NatlWireless.ddf as the parent tree node.)



Creating a DDF for AFP Files

To create a new DDF for an Advanced Function Printing (AFP) file:

1 Select **File>New**. The Configuration Wizard 1 of 4 appears.

2 Specify the project name. The project name cannot be more than 16 characters, start with an alpha character, and should not contain spaces or special characters.

3 Specify the project location or click **Browse** and select a location. The default location is the *Samples* directory.

4 Specify the data file name and path or click **Browse** and select a path.

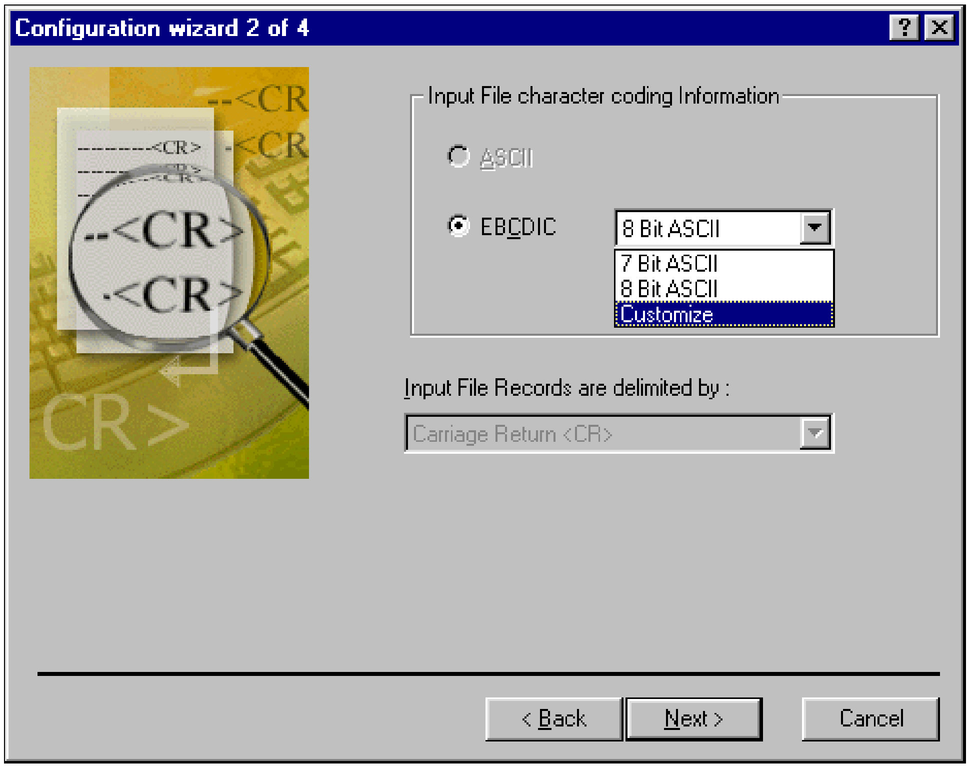
5 Select **AFP** as the file format.

6 Select the country or region. The country/region is important for comparing currency fields, as the currency format differs for different locales.

7 Select or type the name of a character set used in the input file. DefTool cannot verify a typed set name; type carefully and be sure to enter the name of a valid character set.

8 To enable the system to generate Element IDs for unique line item identification, click the Generate Element IDs box. See page 163 for more information about using Element IDs.

9 Click **Next**. The Configuration Wizard 2 of 4 appears.

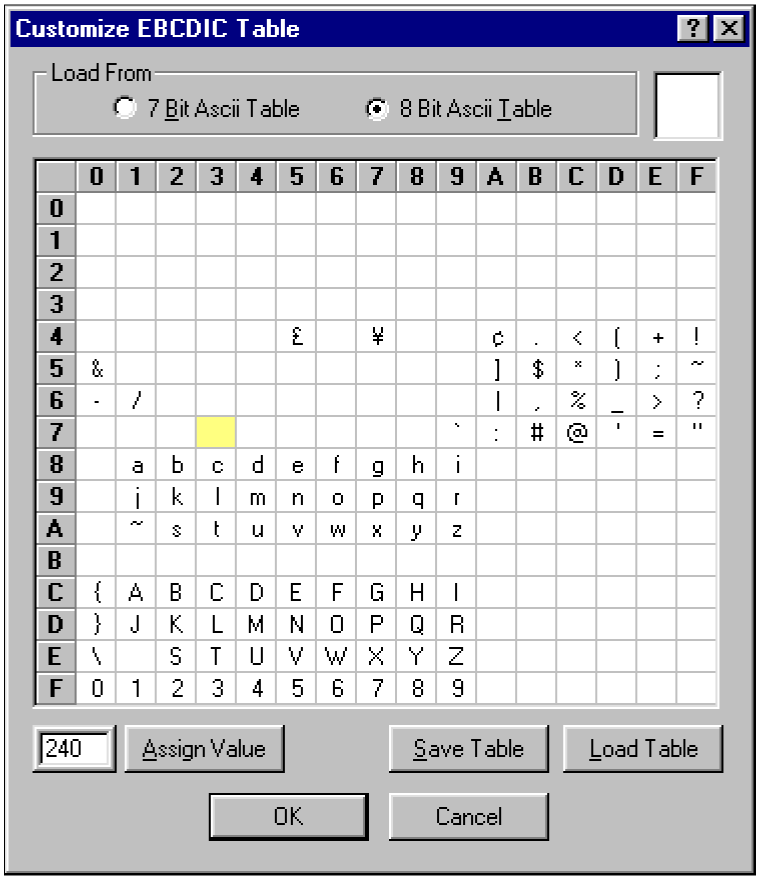

Configuration wizard 2 of 4
Input File character coding Information
ASCII
EBCDIC
8 Bit ASCII
7 Bit ASCII
8 Bit ASCII
Customize
Input File Records are delimited by :
Carriage Return <CR>
< Back
Next >
Cancel

10 Specify the input file character coding. For AFP this must be EBCDIC.

11 Select **`7-bit ASCII`**, **`8-bit ASCII,`** or **`Customize`** to convert EBCDIC characters into ASCII.

If you want to support Extended Characters, you have to create a customized map of EBCDIC to ASCII, which is then saved in the DDF. DefTool also allows you to create a map and save it to an external file, which can be import into DefTool. You can create a single map for every language, and then import that map for use with any DDF using that language.

12 If you select Customize, the Customize EBCDIC Table dialog box appears.



Customize EBCDIC Table

Load From: ( ) 7 Bit Ascii Table [✓ 8 Bit Ascii Table]

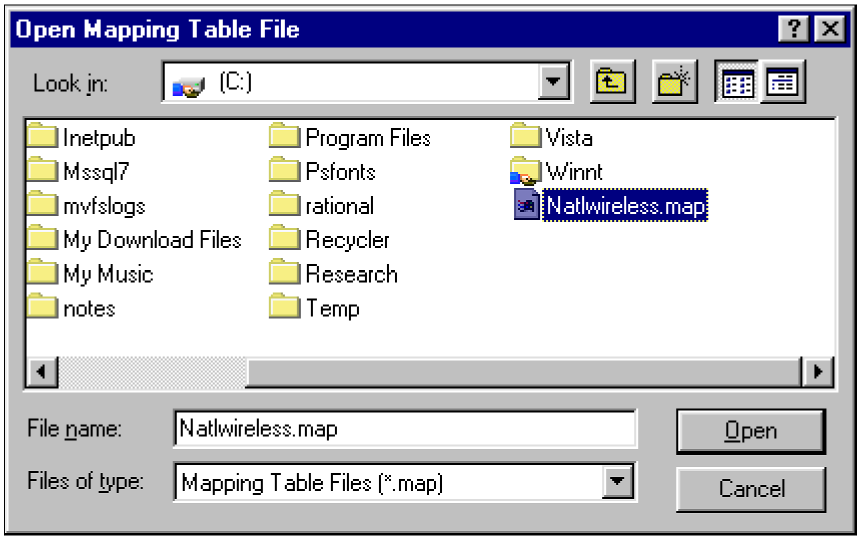
	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0																
1																
2																
3																
4						£		¥			¢	.	<	(	+	!
5	&										]	$	*	)	;	~
6	-	/									|	,	%	_	>	?
7										`	:	#	@	'	=	"
8		a	b	c	d	e	f	g	h	i						
9		j	k	l	m	n	o	p	q	r						
A		~	s	t	u	v	w	x	y	z						
B																
C	{	A	B	C	D	E	F	G	H	I						
D	}	J	K	L	M	N	O	P	Q	R						
E	\		S	T	U	V	W	X	Y	Z						
F	0	1	2	3	4	5	6	7	8	9						

240 | Assign Value | Save Table | Load Table

OK | Cancel

13 Select either the 7-bit EDCDIC table or the 8-bit EBCDIC table as the table from which to load the characters.

If you want to load an existing map, click Load Table and then click OK.



Open Mapping Table File

Look in: (C:)

Inetpub	Program Files	Vista
Mssql7	Psfonts	Winnt
mvfslogs	rational	Natlwireless.map
My Download Files	Recycler	
My Music	Research	
notes	Temp	

File name: Natlwireless.map

Files of type: Mapping Table Files (*.map)

Open | Cancel

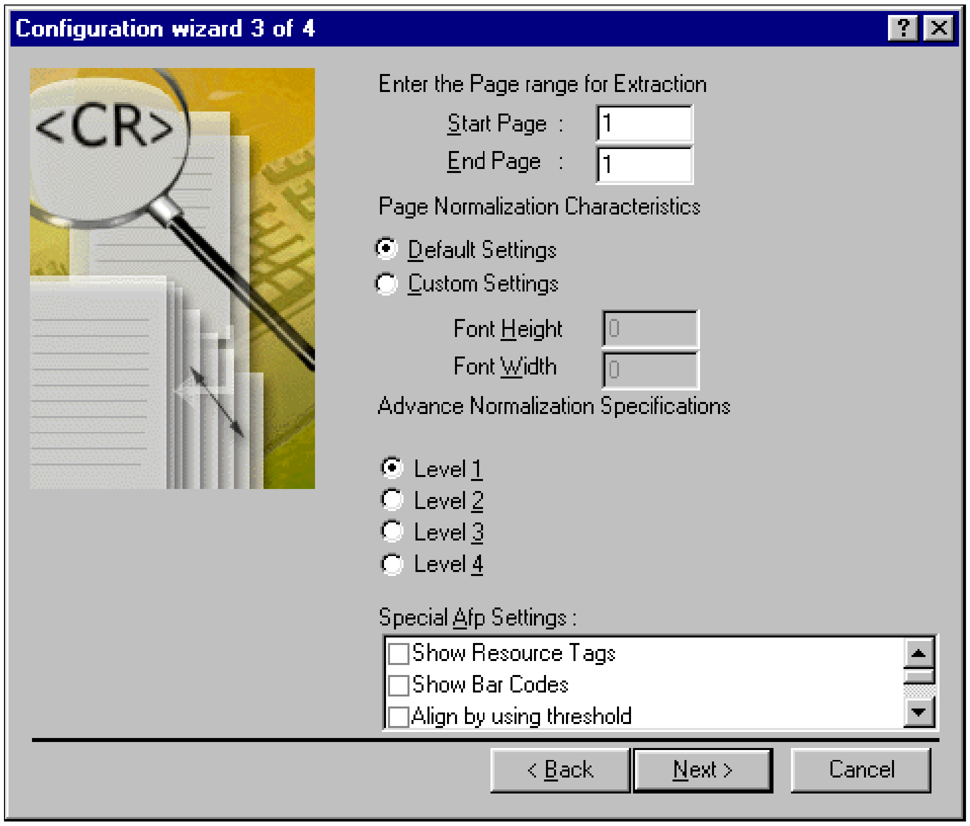
14 In the grid, select the location of the EBCDIC value that should be replaced by the Extended Character. The location will be highlighted. Enter the value for the character in hexadecimal in the data box next to Assign Value.

15 Click **`Assign Value`**.

16 To save the map to an external file, select **`Save Table`**.

17 Click **`OK`** to return to the Wizard 2 of 4.

18 Click **`Next`**. The Configuration Wizard 3 of 4 appears.


Configuration wizard 3 of 4

Enter the Page range for Extraction

Start Page : 1

End Page : 1

Page Normalization Characteristics

(●) Default Settings

( ) Custom Settings

Font Height 0

Font Width 0

Advance Normalization Specifications

(●) Level 1

( ) Level 2

( ) Level 3

( ) Level 4

Special Afp Settings :

☐ Show Resource Tags

☐ Show Bar Codes

☐ Align by using threshold

19 Specify the page range for extraction. It is not required that the start page be 1. For example, the page range can be 5-100. A large range, however, will impact simulation run times. Make sure that the selected range includes a representative sample of data.

20 Select either **`Default`** or **`Custom Setting`** as the page normalization characteristic.

The page normalization characteristics determine how the data file will be displayed in DefTool. DefTool assesses the data elements in the data file and makes a “guess” for presentment. It is better to select **`Default`** and preview the results, and then make any necessary changes as needed.

If **`Custom`** setting is selected, the font height and width must be specified. The font height and width refer to the spacing used when rendering data within DefTool, not the size of font displayed in the window. When you specify a value of one for width, you are telling DefTool to increase horizontal spacing by a factor of one. For each blank space encountered across, DefTool will add an additional space with this setting. When you specify a value for Font Height, you are telling DefTool to add additional vertical spaces at the factor indicated.

The maximum font height for an AFP printer is 72 points (1”) which translates out to 240pel, 300pel, 480pel or 600pel depending on the printer’s resolution. There is no limit to the font width - but it will usually be in direct proportion to the character represented. The only exception to this “rule of thumb” is when a font character is used for a company logo or similar image. In these situations, the font characters can be 1” tall by 2” or 3” wide.

For more information, see the subsequent section, “AFP Font Settings.”

21 The **`Advanced Normalization Settings`** determine the level of processing necessary to display the pages. Select one of the following as the advanced normalization settings:

- **`Level 1`** — Select this option for simple AFP data files where there is no overlapping data. You will not be able to specify the font height and width.
- **`Level 2`** — Select this option for complex AFP data files where there is overlapping data. Level 2 is not as efficient as Level 1 and Level 3.
- **`Level 3`** — A combination of Level 1 and Level 2. This is more efficient than Level 2.
- **`Level 4`** — Level 4 provides the processing capabilities of Level 1, Level 2, and Level 3. It also provides Coded Font Support and BI tag extraction. Level 4 is more efficient than Level 3. Select Level 4 for all new DDFs based on AFP.

If you select Level 4 settings, the Coded Font Support settings display.

22 If you want to support coded fonts, enable the **`Coded Font Support`** check box and specify the Font File path. This can be any directory, but it must contain the following files: *icoded.fnt*, *CPDef.fnt and CSDef.fnt*. Also specify the font height controller and the font width controller.

23 Select any of the following special AFP settings:

- **`Show Resource Tags`** — Displays image and overlay data in the header.
- **`Show Bar Codes`** — Shows bar code tags when displaying the data file in DefTool.
- **`Align by using threshold`** — Removes any unnecessary space lines and aligns the data when displaying the data file in DefTool.
- **`Extract No Operation (NOP)`** from input file — Extracts No Operation (NOP) data from the data file.
- **`Extract Tag Logical Elements(TLE) from input file`** — Any Tag Logical Elements (TLE) in the data file are extracted and presented in DefTool.
- **`Process Horizontal Data Only`** — eStatement Manager only processes the data that is horizontal in the AFP data file. Any other orientation of text (vertical, at any other angle) will not be processed.
- **`Extract BI Tags`** — BI overlay tags in the data file are presented in DefTool. This option is available only if you select Level 4 as the Advanced Normalization Settings.
- **`Use Actual Page Size –`** `Use actual page size defined in the AFP page itself`. This option is available only if you select Level 4 as the Advanced Normalization Settings.

24 Click **`Next`**. The Configuration Wizard 4 of 4 appears. All the configuration settings that were selected will be displayed along with a preview of the data file.

25 If the page formatting is correct, click **Finish** to open the data file in the Definition Tool; otherwise click the **Back** button to go back and change the configuration information.

AFP Font Settings

The DefTool parser takes the structured field that defines:

- The units of measure for the coordinate system (such as 240 dpi or 300 dpi)
- The page size (such as 8.5 X 11 inches)

It then computes the actual horizontal and vertical coordinate space size. For example, if it uses 240 dpi, it means a coordinate space of 2040 dots in the X direction and 2640 dots in the Y direction.

Source files, for example AFP and Metacode files, usually use a high-resolution coordinate system with many units per horizontal and vertical inch. The DefTool must translate the values to the much lower-resolution coordinate system of a computer screen, with many fewer units per horizontal and vertical inch.

This allows the DefTool to:

- Easily display the data inside one "screen" of information displayed in its window.
- Easily display the data inside a standard browser window.

The DefTool makes some assumptions about how to do the translation from a high-resolution system to a low-resolution system. The Width and Height settings allow you to increase the horizontal and vertical resolution of the grid system used in the DefTool. The default setting is 0,0, and changing these settings will increase the resolution of either the horizontal (Width) or vertical (Height) grid unit system in the DefTool.

The following describes a case where you would need to use character Width and Height information to successfully map AFP data to a text display inside the DefTool.

The X and Y coordinates where a text string starts on the page are extracted from the AFP composed text record, and these coordinates are translated into DefTool coordinates. For example, assume that the X, Y coordinate for the text string "eaDirect" is 120,200. Next, assume that the DefTool automatically scales down these coordinates to a coordinate system that has only 1/10th the number of units in both the X and Y directions. That means those coordinates are translated to 12,20.

This implies that for every 10 X direction coordinate units in the print document, there is only one coordinate position available in the X direction in the DefTool. For every 20 Y direction coordinate units in the print world, there is only one coordinate position available in the Y direction. It also means that it may encounter rounding issues: X direction print coordinates 1 to 10 are mapped to DefTool character position 1, X direction print coordinates 11 to 20 are mapped to DefTool character position 2, etc.

When the DefTool extracts data from the text string, it places each character in sequence into the DefTool display coordinate system, starting at the X coordinate calculated (12, in this case). For example, the DefTool places the 10 characters in the text string "eaDirect" into positions 12 through 21, accommodating the 10 characters in the string.

This system works well until the DefTool coordinate system induces too much "rounding error" into the resolution reduction process. For example, the DefTool often processes print files with extremely small font widths. If the characters in the word "eaDirect" were rendered using printer font characters that were only six dots wide, the total horizontal coordinate space occupied on the printed page would be 10 characters X 6 dots, or 60 total dots.

When you add this to the starting X position of 120, you have the string "eaDirect" occupying printer coordinate positions 120 through 180. The DefTool uses the resolution already defined for the X direction (1 unit or every 10 units of horizontal resolution in the print file), to place the characters out to position 21.

Now assume that the print file contains a text string at position 185,200. That places the text string 5 dots to the right of the end of the string "eaDirect" on the printed page ("eaDirect" ends at position 180). DefTool maps the X coordinate of 185 in the print document by dividing 185 by 10, or 18.5, which it rounds up to position 19. The DefTool will map anything from X position 181 to 190 in the print file to X position 19. However, it already placed a character - the "e" in "eaDirect" into this character position inside the DefTool. This is how character "overlap" can occur in the DefTool.

This problem is the result of too great a reduction in the X direction resolution when the DefTool converting from printer X direction coordinates to DefTool X direction coordinates using the current Font Width extraction value. You can adjust this value to increase or decrease the number of X direction printer units mapped to X direction DefTool units.

Creating a DDF for DB Extract Files

To create a new DDF for a DB extract data input file:

1 Select **File>New**. The Configuration Wizard 1 of 5 appears.

2 Specify the project name. The project end cannot be more than 16 characters and should not contain spaces or special characters.

3 Specify the project location or click **Browse** and select a location. The default location is the Samples directory.

4 Specify the data file name and path or click **Browse** and select a path.

5 Select **DBExtract** as the file format.

6 Select the country or region. The country/region is important for comparing currency fields, as the currency format differs for different locales.

7 Select or type the name of a character set used in the input file. DefTool cannot verify a typed set name; type carefully and be sure to enter the name of a valid character set.

8 To enable the system to generate Element IDs for unique line item identification, click the Generate Element IDs box. See page 163 for more information about using Element IDs.

9 Click **Next**. The Configuration Wizard 2 of 5 appears.

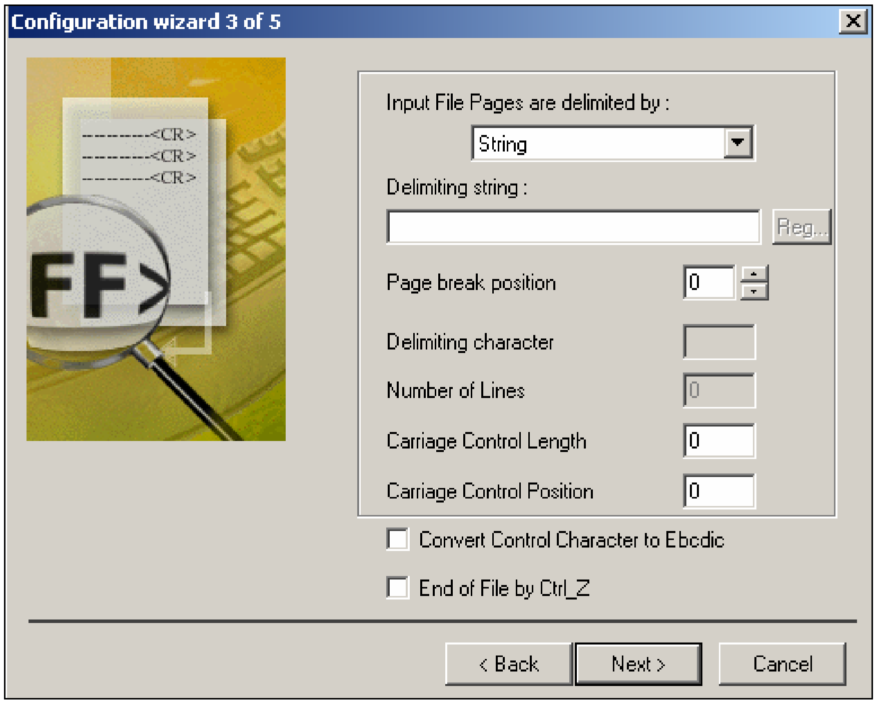
10 The input file character coding for BD Extract can be either **ASCII** or **EBCIDIC**.

11 Specify how the tables are delimited in the input file. Currently the valid options are:

- **Carriage Return (CR)** — A control character that moves the printer head to the start of the next line, indicating the end of a table.
- **Line Feed (LF)** — A control character that moves the printer head down one line indicates the end of a table.

- **`CR and / LF`** — Table break is identified by both a <CR> and <LF> or either one.
- **`Number of Characters`** — The table ends after a fixed number of characters.
- **`Record Split`** — Use this option to split large Records. You must specify the number of fields in each line.

12 Click **`Next`**. The Configuration Wizard 3 of 5 appears.



Input File Pages are delimited by : String

Delimiting string :

Page break position 0

Delimiting character

Number of Lines 0

Carriage Control Length 0

Carriage Control Position 0

☐ Convert Control Character to Ebcdic

☐ End of File by Ctrl_Z

13 Specify how the input file page breaks are delimited. The valid options are:

- **`Form Feed`**
- **`Carriage Return by Position`**
- **`Carriage Return by Length`**
- **`Number of Lines`**
- **`Character`**
- **`String`**
- **`Regular Expression (`**a changing string of characters that follow a logical pattern)

14 If you selected String as the page break delimiter, specify the exact string in the Delimiting String field. Enter the starting column position for the string in the Page Break Position field (if not 0).

15 If you selected Regular Expression as the page delimiter, click Reg... and use the Regular Expression dialog to build the identifying expression.

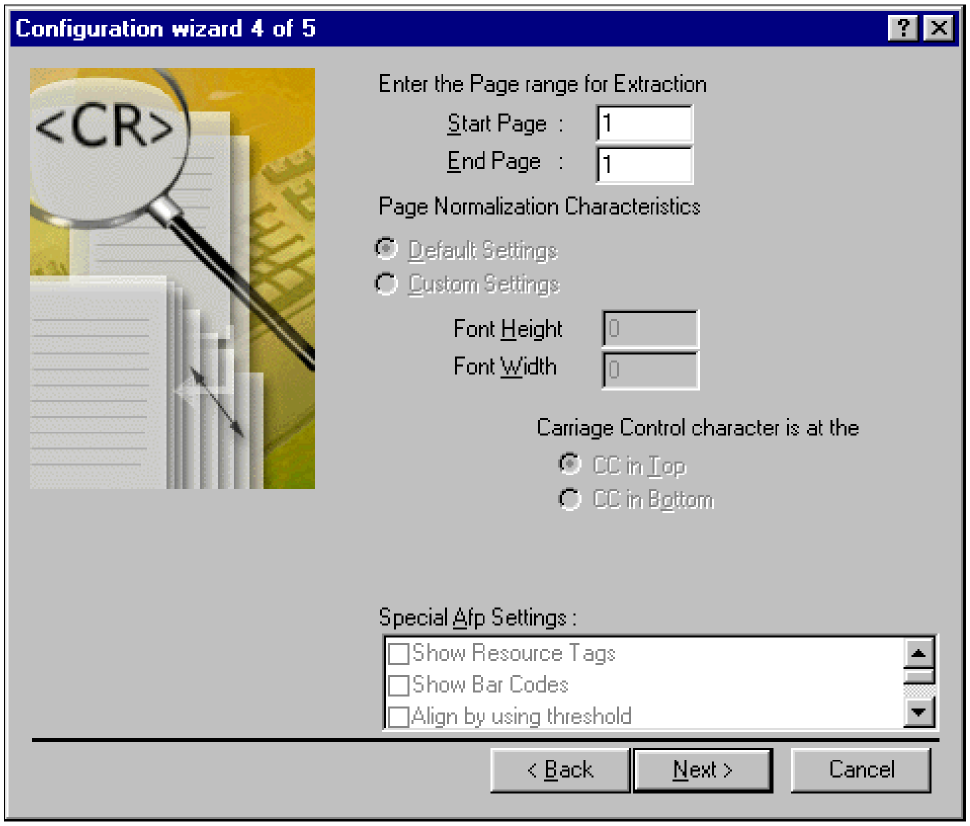
16 If you selected Character as the page delimiter, identify the character in the Delimiting character field.

17 If you selected a Number of Lines per page denotes page breaks, specify the number of lines per page in the Number of Lines field.

18 If you selected Character or String as the page delimiter, select the **`Convert Control Character to EBCDIC`** option if the only way of entering the delimiting character is to specify its ASCII value and then internally convert it to EBCDIC.

19 If necessary, select the **`End of File by Ctrl_Z`** option. The default end of file character is **`0xFF`**, but some data files can have **`Ctrl Z`** as the end of file control character.

20 Click **`Next`**. The Configuration Wizard 4 of 5 appears.



Enter the Page range for Extraction

Start Page : 1

End Page : 1

Page Normalization Characteristics

Default Settings

Custom Settings

Font Height 0

Font Width 0

Carriage Control character is at the

CC in Top

CC in Bottom

Special Afp Settings :

Show Resource Tags

Show Bar Codes

Align by using threshold

21 Specify the page range for extraction. It is not required that the start page be 1. For example the page range can be 5-100. A large range, however, will impact simulation run times. Make sure that the selected range includes a representative sample of data.

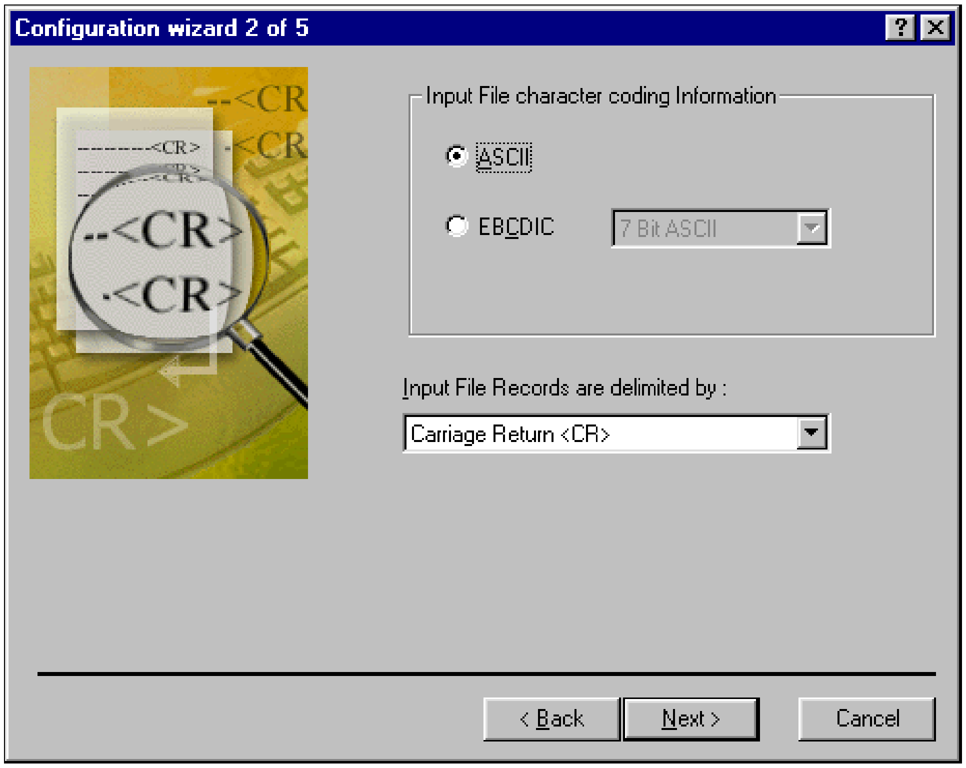
22 Click **`Next`**. The Configuration Wizard 5 of 5 appears. All the configuration settings that were selected will be displayed along with a preview of the data file.

23 If the page formatting is correct, click **`Finish`** to open the data file in the Definition Tool; otherwise click the **`Back`** button to go back and change the configuration information.

Creating a DDF for Line Printer Format Files

To create a new DDF for a line printer data input file:

1. Select **`File>New`**. The Configuration Wizard 1 of 5 appears.
2. Specify the project name. The project name cannot be more than 16 characters and should not contain spaces or special characters.
3. Specify the project location or click **`Browse`** and select a location. The default location is the Samples directory.
4. Specify the data file name and path or click **`Browse`** and select a path.
5. Select **`Line Printer`** as the file format.
6. Select the country or region. The country/region is important for comparing currency fields, as the currency format differs for different locales.
7. Select or type the name of a character set used in the input file. DefTool cannot verify a typed set name; type carefully and be sure to enter the name of a valid character set.
8. To enable the system to generate Element IDs for unique line item identification, click the Generate Element IDs box. See page 163 for more information about using Element IDs.
9. Click **`Next`**. The Configuration Wizard 2 of 5 appears.


Configuration wizard 2 of 5

Input File character coding Information

(● ASCII) ○ EBCDIC 7 Bit ASCII

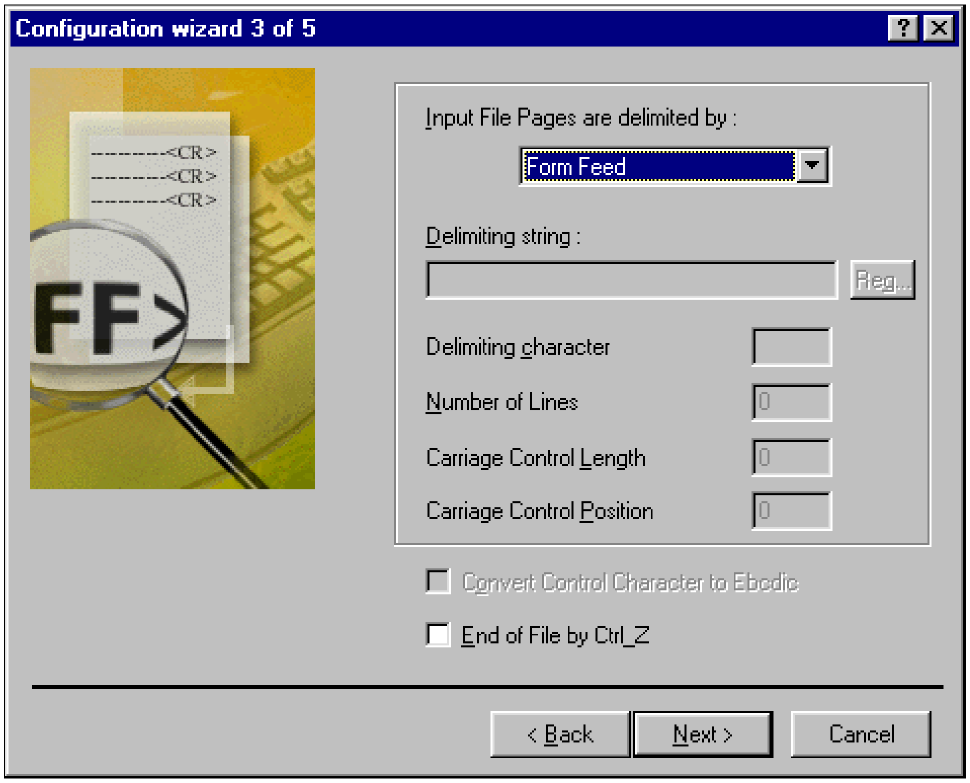
Input File Records are delimited by : Carriage Return <CR>

10. The default input file character coding for Line Printer is **`ASCII`**.

11 Specify how the records are delimited in the input file. Currently the valid options are:

- **`Carriage Return (CR)`** — A control character that moves the printer head to the start of the next line, indicating the end of a record.
- **`Line Feed (LF)`** — A control character that moves the printer head down one line indicates the end of a record.
- **`CR and / LF`** — Record break is identified by both a <CR> and <LF> or either one.

12 Click **`Next`**. The Configuration Wizard 3 of 5 appears.



Input File Pages are delimited by : Form Feed

Delimiting string :

Delimiting character

Number of Lines 0

Carriage Control Length 0

Carriage Control Position 0

☐ Convert Control Character to Ebcdic

☐ End of File by Ctrl_Z

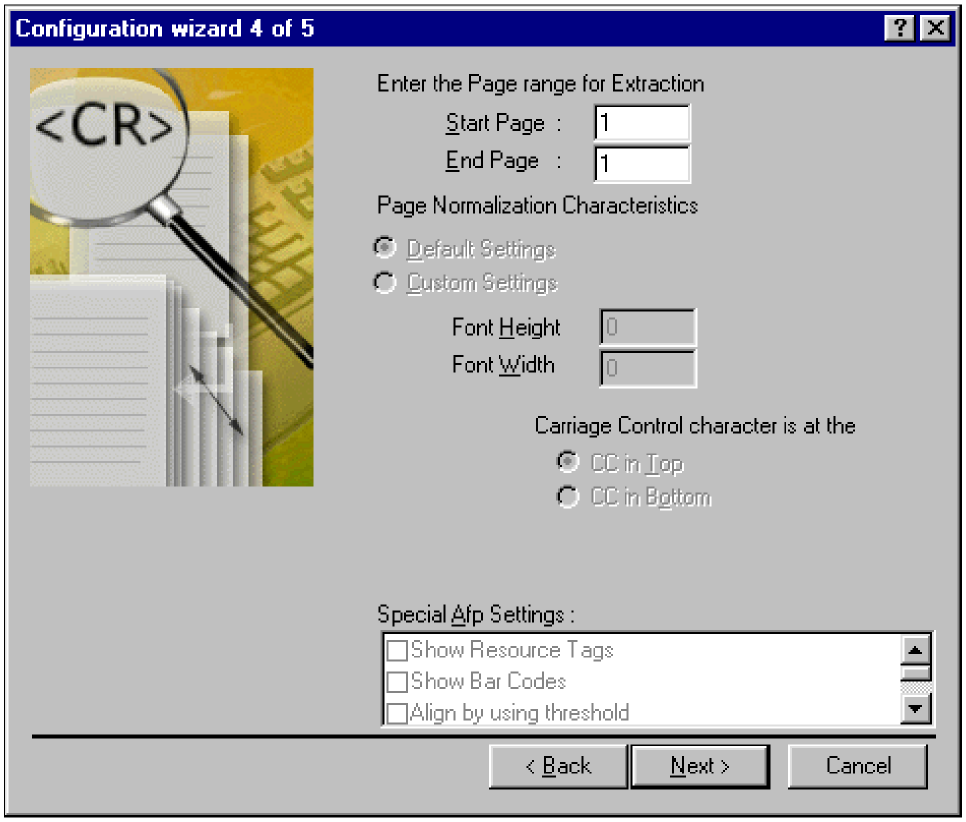
13 Specify how the input file pages are delimited. Currently the valid options are:

- **`Form Feed`** — This control character directs the printer to advance the paper to the top of the next page. For example, a Form Feed can indicate the end of a page.
- **`Character`** — A specific character denotes the Page breaks. You must specify the character.
- **`String`** — A specific string identifies the Page breaks. You must specify the exact string.

14 Check the **`Convert Control Character to EBCDIC`** check box, if the only way of entering the delimiting character is to specify its ASCII value and then internally convert it to EBCDIC. This option is available only if you select **`Character`** or **`String`** as the page delimiter.

15 If necessary, check the **`End of File by Ctrl_Z`** check box. The default end of file character is **`0xFF`**, but some data files can have **`Ctrl Z`** as the end of file control character.

16 Click **`Next`**. The Configuration Wizard 4 of 5 appears.



Enter the Page range for Extraction

Start Page : 1

End Page : 1

Page Normalization Characteristics

(● Default Settings) ○ Custom Settings

Font Height 0

Font Width 0

Carriage Control character is at the

(● CC in Top) ○ CC in Bottom

Special Afp Settings :

☐ Show Resource Tags
☐ Show Bar Codes
☐ Align by using threshold

17 Specify the page range for extraction. It is not required that the start page be 1. For example the page range can be 5-100. A large range, however, will impact simulation run times. Make sure that the selected range includes a representative sample of data.

18 Click **Next**. The Configuration Wizard 5 of 5 appears.

Creating a DDF for Metacode Files

To create a new DDF for a Metacode file:

1 Select **File>New**. The Configuration Wizard 1 of 5 appears.

2 Specify the project name. The project end cannot be more than 16 characters and should not contain spaces or special characters.

3 Specify the project location or click **Browse** and select a location. The default location is the *Samples* directory.

4 Specify the data file name and path or click **Browse** and select a path.

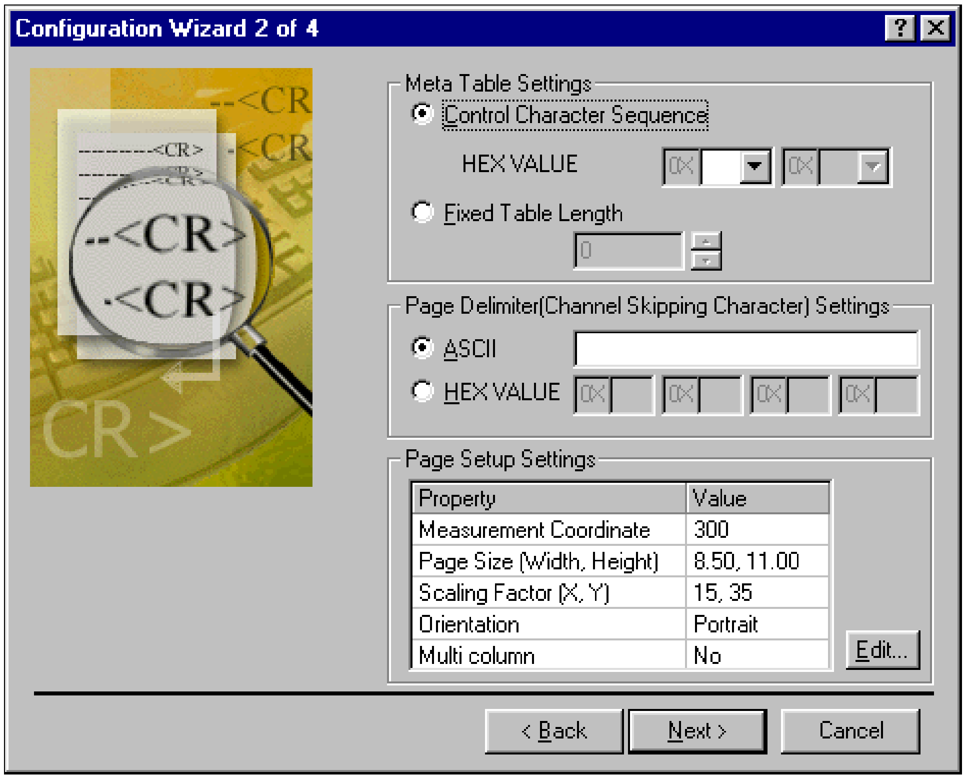
5 Select **Metacode** as the file format.

6 Select the country or region. The country/region is important for comparing currency fields, as the currency format differs for different locales.

7 Select or type the name of a character set used in the input file. DefTool cannot verify a typed set name; type carefully and be sure to enter the name of a valid character set.

8 To enable the system to generate Element IDs for unique line item identification, click the Generate Element IDs box. See page 163 for more information about using Element IDs.

9 Click **`Next`**. The Configuration Wizard 2 of 4 appears.



Property	Value
Measurement Coordinate	300
Page Size (Width, Height)	8.50, 11.00
Scaling Factor (X, Y)	15, 35
Orientation	Portrait
Multi column	No

10 Specify how the records are delimited in the input file. Currently the valid options are:

- **`Control Character Sequence`** — A sequence of control characters signals the end of a record. This is normally given in hexadecimal. You can select commonly used values from the combo box.

- **`Fixed Record Length`** — Records have a fixed number of bytes. You have to specify the number of bytes.

11 Specify how the input file pages are delimited (Channel Skipping Character). Currently the valid options are:

- **`ASCII`** — Specific ASCII characters denote the Page breaks. You must specify the characters. The length is limited to four values.

- **`Hexadecimal`** — Page breaks are identified by specific Hexadecimal values. You have to specify the values. The length is limited to four values.

12 Click **`Edit`** to specify the page setup settings. The Page Setup dialog box appears.

13 Specify the **`measurement coordinates`**. This can either be 300 dpi or 240 dpi. The default is 300 dpi.

14 Specify the **`page size`**. Currently the valid options are:

- **`Letter`** — 8.5 x 11.0 inches
- **`A4`** — 8.27 x 11.69 inches
- **`Custom`** — You have to specify the page width and height.

15 Specify the **`X`** and **`Y scaling factors`**. This identifies the scaling factor used to display the file in DefTool. X specifies the number of pixels per column and Y specifies the number of pixels per row.

16 Select **`Portrait`**, **`Landscape`** or **`Portrait & Landscape`** as the orientation. Some Metacode files have data in both Portrait and Landscape format, so you will have to select Portrait & Landscape for these files.

17 If necessary, enable the **`Multi column`** check box, to allow multiple columns of data to be displayed in DefTool. The Right Column Properties section will be enabled in the Page Setup dialog box. These settings depend on the page orientation you select. In the screen shot shown below, Portrait and Landscape are selected as the page orientation. As a result the **`Right Column Properties`** for both are enabled. If only one orientation is selected, then only the relevant section will be enabled.

18 Specify the column break position. This indicates the end of the first column to the left. Normally this would be the first blank space after the first column.

19 Specify how to shift the column. i.e. Shift right or shift down. Also specify the number of columns by which to shift the next column. If you select Shift Down and do not specify the number of columns to shift by, zero will be taken as the starting column.

If you need to go back to the default page settings, click Default. The default settings are: Measurement coordinate –300 dpi, Page Size - Letter (8.5*11.0 inches), Scaling Factor - X = 15 and Y = 35, Orientation - Portrait

20 Click **`OK`** to return to Configuration Wizard 2 of 4. The specified page setup settings will be displayed.

21 Click **`Next`**. The Configuration Wizard 3 of 4 appears.

22 Specify the **`page range`** for extraction. It is not required that the start page be 1. For example the page range can be 5-100. A large range, however, will impact simulation run times. Make sure that the selected range includes a representative sample of data.

23 Click **`Next`**. The Configuration Wizard 4 of 4 appears. All the configuration settings that were selected will be displayed along with a preview of the data file.

24 If the page formatting is correct, click **`Finish`** to open the data file in the Definition Tool; otherwise click the **`Back`** button to go back and change the configuration information.

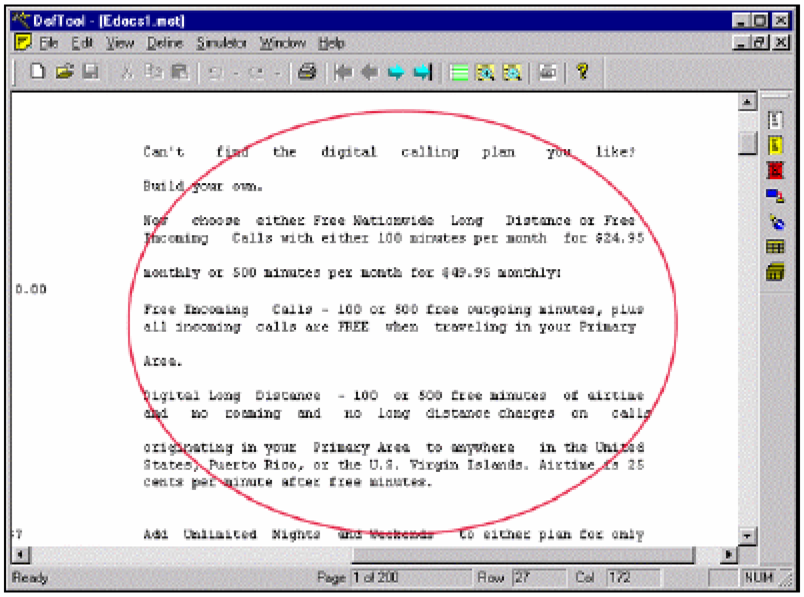
Metacode Settings

X and Y Scaling Factors

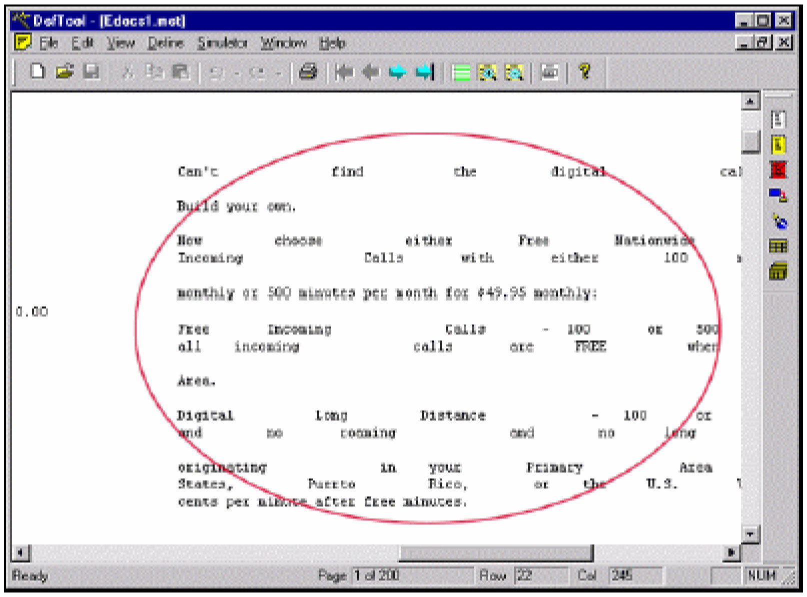
Print files specify information in pixels, but the DefTool requires row and column positions.

X Scaling factor — The number of pixels per column or the average width of each character. The default X scaling factor is 15 pixels. Decreasing the X scaling factor will introduce unnecessary spaces between the words.

Increasing the X scaling factor will reduce the space. The images below show how the X scaling factor changes how the DefTool displays data. In the first image, the X scaling factor is 17, while in the second image the X scaling factor is eight, which introduces more spaces between words.



```
          Can't   find   the   digital   calling   plan   you   like?

          Build your own.

          Now   choose   either Free Nationwide  Long   Distance or Free
          Incoming   Calls with either 100 minutes per month  for $24.95

          monthly or 500 minutes per month for $49.95 monthly:
0.00
          Free Incoming   Calls - 100 or 500 free outgoing minutes, plus
          all incoming  calls are FREE  when  traveling in your Primary

          Area.

          Digital Long  Distance  - 100  or 500 free minutes  of airtime
          and  no roaming and  no long  distance charges on  calls

          originating in your  Primary Area  to anywhere  in the United
          States, Puerto Rico, or the U.S. Virgin Islands. Airtime is 25
          cents per minute after free minutes.

:7        Add  Unlimited  Nights  and Weekends  to either plan for only
```



```
          Can't            find          the          digital           cal

          Build your own.

          Now        choose         either        Free        Nationwide
          Incoming                Calls       with        either       100      a

          monthly or 500 minutes per month for $49.95 monthly:
0.00
          Free       Incoming                Calls       -  100       or    500
          all    incoming              calls       are     FREE           when

          Area.

          Digital           Long          Distance                -   100        or
          and         no       roaming                and        no       long

          originating                in    your        Primary           Area
          States,          Puerto           Rico,        or    the      U.S.       V
          cents per minute after free minutes.
```

Y Scaling Factor — The number of pixels per row or the average height of each character. The default Y scaling factor is 35 pixels. Decreasing the Y scaling factor will introduce unnecessary spaces between two rows.

Column Break Position

The column break position gives the start of the second column. (An approximate column value is sufficient.) Before setting this value, open the file in DefTool and make sure that the page appearance is similar to the printed-paper document. Once you get the correct output in DefTool, you must then define rules for extracting your data.

In the data file shown in the images below, there was data in both Landscape and Portrait orientations, the X scaling factor was 17 and the Y scaling factor was 35. With the selected scaling factors in Portrait orientation, some of the data that should be in column 1 is displayed in column 2.

In order to display the data correctly, the second column must shift to the right. Here it has been shifted 20 positions to the right.

TIP: You can also shift the second column down, in which case the second column will appear below the first column. If you shift the second column down, then you have to specify the starting position. The default start column position is zero.

5 Writing Data Extraction Rules

What are Data Extraction Rules?

An extraction rule is what you write when you define a document style, primary key, additional page style, field, table, or group in DefTool.

Each of these rules identifies a unique, individual entity on a statement in the data input file.

Some extraction rules, such as field extraction rules, tell eStatement Manager where to search on a statement page and how to find data to extract to display it in an online statement or to use it with conditional business logic. We write other rules, such as the document style, for the purpose of telling eStatement Manager how to identify new statements within the data input file.

You must create well-defined rules that can successfully identify and extract these values. This requires an understanding of the data and all its possible forms, including its unique properties, so that you create a data pattern to use for extraction.

This section describes the general process of creating extraction rules.

Identifying Unique Properties of Data to be Extracted

For each statement element you want to extract, you must identify:

- The data's unique **location** in the DefTool search area.
- The unique **content** properties or characteristics of the data values that can be used in rules definition.

Search Area (Location)

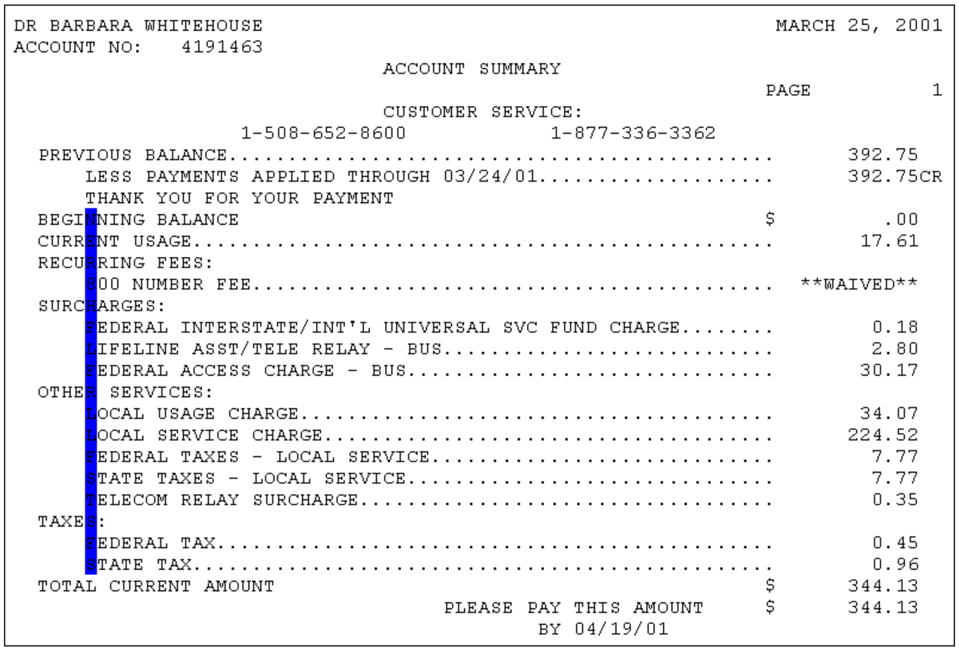
Where does the data appear within the data source? As you graphically select your values in the work area to begin creating the rules to extract them, you need an understanding of where the data can appear on all the different types of statements that exist within each data source. You also need an understanding of the variability within each statement type. The window coordinates that you define (starting with the process of highlighting an area with the mouse, and refined within the Properties and Information dialogs) identify where the production components should look for the start of the data string.

An area, defined via the "grid" that has been applied to each statement page, of columns and rows that indicates where the START of the data string should be searched. This area is called a window coordinate set. The window coordinate set that you define should be as small as possible, defining the minimum search area necessary to locate the first position of the data string specified. This is because the eStatement Manager extraction components actually conduct the search for data from left to right, top to bottom.

It is often necessary to create window coordinate sets that span many rows on a page, for example. If you define a data string to be searched for somewhere between row 3 and row 100, then the extraction components will look on every row from row 3 until the correct data string is found. Assume, then, that the data string will always be found between column 40 and column 41. This means that your window coordinate set should not be any wider than that one expected column set. Expanding the set even by one column, i.e. to column 42, increases the searching area two-fold. Excessively large window coordinate sets will slow down the performance of live retrieval during deployment.

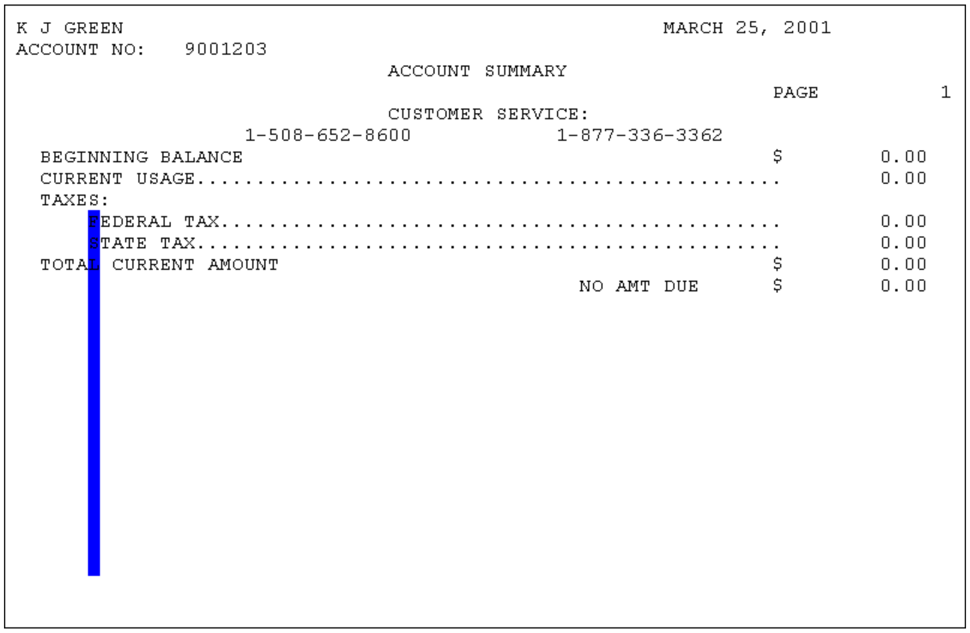
Example

A customer uses several of your services. Due to the detail in the statement, the Federal and State Taxes will appear near the bottom of the page (illustration below).



```
DR BARBARA WHITEHOUSE                                                MARCH 25, 2001
ACCOUNT NO:    4191463
                              ACCOUNT SUMMARY
                                                                    PAGE           1
                              CUSTOMER SERVICE:
                   1-508-652-8600            1-877-336-3362
  PREVIOUS BALANCE.............................................       392.75
      LESS PAYMENTS APPLIED THROUGH 03/24/01....................      392.75CR
      THANK YOU FOR YOUR PAYMENT
  BEGINNING BALANCE                                            $         .00
  CURRENT USAGE................................................        17.61
  RECURRING FEES:
      800 NUMBER FEE...........................................    **WAIVED**
  SURCHARGES:
      FEDERAL INTERSTATE/INT'L UNIVERSAL SVC FUND CHARGE........        0.18
      LIFELINE ASST/TELE RELAY - BUS............................        2.80
      FEDERAL ACCESS CHARGE - BUS...............................       30.17
  OTHER SERVICES:
      LOCAL USAGE CHARGE........................................       34.07
      LOCAL SERVICE CHARGE......................................      224.52
      FEDERAL TAXES - LOCAL SERVICE.............................        7.77
      STATE TAXES - LOCAL SERVICE...............................        7.77
      TELECOM RELAY SURCHARGE...................................        0.35
  TAXES:
      FEDERAL TAX...............................................        0.45
      STATE TAX.................................................        0.96
  TOTAL CURRENT AMOUNT                                         $      344.13
                                   PLEASE PAY THIS AMOUNT      $      344.13
                                           BY 04/19/01
```

Another customer subscribes to one of your services, or infrequently uses your services, requiring lesser detail in the statement. In this statement, the Federal and State Taxes appear in the upper third of the of the page, as shown below.



```
K J GREEN                                               MARCH 25, 2001
ACCOUNT NO:   9001203
                               ACCOUNT SUMMARY
                                                                 PAGE          1
                               CUSTOMER SERVICE:
                   1-508-652-8600            1-877-336-3362
  BEGINNING BALANCE                                              $        0.00
  CURRENT USAGE................................................           0.00
  TAXES:
      FEDERAL TAX..............................................           0.00
      STATE TAX................................................           0.00
  TOTAL CURRENT AMOUNT                                           $        0.00
                                                NO AMT DUE       $        0.00
```

The value for taxes starts in the same column on all the bills. To successfully extract the taxes on these two types of statements, the Taxes field window needs to span all rows from the upper third to the bottom of the page, yet only needs to span one column. This is indicated by the blue field indicator for the Taxes field.

Data Patterns (Content Search)

eStatement Manager supports extracting data as a string only. For each piece of data you want to extract, you must identify a data pattern that eStatement Manager can use to match any and all occurrences of the data string. Depending on the possible data occurrences, a data pattern can be

- Static data string (such as **`ACCOUNT NO`**:), or
- A variable regular expression (more common)

With a regular expression, you define, for each position of data, what characters are valid for extraction. You may want to extract only data strings that consist of 3 numeric values. In this case, you would write a regular expression that requires each of the 3 positions of data to be numbers, i.e. [0-9][0-9][0-9].

What does the data look like? The knowledge of what characters are possible in each position of the data string, as well as how many characters make up any given string, will enable you to define precise rules that will ultimately extract only the data you want, as you want it to appear.

Example

On Biller A’s statements, the Account Number is a 14-digit numeric string containing a dash after every four digits. Thus, it appears in the data source as `####-####-####`.

The Phone Number is a 10-digit numeric string with dashes that separate the area code, exchange and individual number, respectively. Thus, it appears in the data file as `###-###-####`.

The data on these statements floats slightly from statement to statement and are located near each other. Thus, there is a chance that the window coordinate sets that you define to extract these values will overlap.

To successfully extract these two values without confusion as to what value belongs to what field, you need to specify tightly defined extraction rules that detect the differences in pattern and length.

Field Properties

Fields are data strings that

- Occur on one row only, and
- Do not iterate within a statement; (they have only one value per statement).

When defining the rules to extract data fields, you must specify the following field properties:

Data Type — Data can be classified as a String, Currency or Numeric type. You select the type that is appropriate to the data, keeping in mind your plans for this field during the indexing process. Perhaps you may want to index a field called TotalAmountDue to the application database, for research or marketing reasons. It would make sense to classify TotalAmountDue as a Currency value, so as to ensure that the corresponding database field would also be defined as Currency.

Data type has an effect on field use in Business Logic. You may want to use the TotalAmountDue field in a Business Logic rule that offers your customers a different set of marketing promotions, based on the amount of the current statement. By classifying the TotalAmountDue Data Type as **CURRENCY**, you can be sure of accurate calculations of comparisons using Business Logic.

For example, a TotalAmountDue of $169.48 can be compared to $1000.00. Since the TotalAmountDue is less than $1000, the customer would receive a statement with a less ambitious set of promotions.

Field Type — Specify the field type when creating your primary key only. You use this definition only once, on the first field you create.

Formatting Specifications — For applying post-extraction data conversions to the data. You can apply:

- Alphanumeric conversions, such as ‘Delete all trailing zeros or spaces’
- Date conversions, such as ‘Translate Aug 21, 2002 to 08/21/2002’

Force Field Length — You may want to extract a field by validating a few characters of the string found, and then extracting all other characters to a specified number. This is useful in cases where the pattern of the field can vary widely, such as in a customer name or street address Line. In these cases, the regular expression that you would have to define as the valid pattern might be so broad that it could result in an impact to extraction performance. You could specify that the first character must be alphanumeric (i.e., *[A-Za-z0-9]*), but to extract the subsequent characters until a maximum number of 30 characters have been extracted. This is commensurate to enabling the Force Fixed Length option, and setting the length to 30.

TIP: All indexing fields will normally have the Force Fixed Length attribute engaged.

Marker Properties

For each table and group you want to extract from your data input file, you must create a pair of start and end markers that eStatement Manager uses to locate the start and end of the table or group.

Define the start marker on a data string (variable or static) that always appears on or above the table or group occurrences. Similarly, you must define an end marker on a data string that always appears on or below the table or group occurrences.

When defining the rules to create markers, you must specify the following properties:

Special Naming — The name you choose for your start and end marker must be the same. This identifies them as a marker pair. Also give the table or group associated with the marker the same name as the marker pair; this is a 'best practice' for organizational purposes.

Include Marker Line — Start and end markers can exist on the same row where there may be table or group data (including markers for the tables of a group). If this is the case, you check on the Include Marker Line option so that these rows are also searched for column data, or group information (including the group's table markers and columns).

Use Blank Line as Marker — Sometimes, the best indicator of the start or end of a table or group is not a data string that appears with the table or group instances, but is rather a row in the statement containing no data. In this case, you can check on the Use Blank Line as Marker option.

Select the Line Only Containing This Pattern — If you are sure that the data string you are using as your marker is the only data on its line, you should enable this option. This requires that NO other characters appear between the rows that contain the marker string.

Look for the Start Marker on the Same Row — This option is useful when your start and end marker patterns are the same and expected to be found in the same location. Say, for example, that your start marker is string X in location Y, and your end marker is string X in the same location Y, yet is expected on a subsequent page. You would then enable this option in order to force the extraction components to search beyond the first occurrence of string X, thus avoiding the creation of a null search area for the data.

Use Dynamic Pattern — This option will allow you to insert a dynamic field in the marker's pattern. When this option is enabled, you will be given a dropdown list to select the dynamic field for the pattern, or part of the pattern.

Table Properties

Tables consist of one or more related data strings that you want to extract and present together. These data strings may have multiple occurrences. The data strings that you extract as part of a table are defined as columns.

Table Anchors

Every table you create must also contain an anchor definition. The table anchor is a data string that occurs with every occurrence of the table columns. If the table anchor pattern is not found where it is looked for on a row between the start and end marker, then this row becomes invalid for the search of the table columns. Essentially, the table anchor occurrences MUST appear on every line that there could be table data. The two options for your table anchor are:

- **Internal Anchor** — This is a table column that you expect to always be populated with data. For example, in a call itemization list, the charge column would be populated with data for every List Item. Therefore, you would select this column to be the anchor for itself and the remaining columns. By default, the table uses the first defined column as the Internal anchor.
- **External Anchor** — If you are not certain that one of your table columns will always be populated with data, then it may be necessary to define a data string that is external to the table itself as the anchor. A good example of external anchor data would be control codes that may exist with on the periphery of every table line. You can only define an external anchor if the appropriate data exists on every row of table column data.

Table Columns

You define the rules to extract table columns similarly to the same way you define a field. A major difference is in the window coordinate set that you define. When you are creating columns of a table, it is important to keep the following points in mind:

- Row coordinates of columns should only span one row. The rows that will ultimately be searched for column data will be defined dynamically by the detection of the table start and end markers during extraction. Therefore, you **do not** need to anticipate the row span of your columns when you are creating their rules. You only need to select one row (i.e. start row = 10 and end row = 11) when creating your window coordinate set.
- In most cases, choose the same row span for all columns in a table. More specifically, the row span that you choose for your columns must correspond to the visual pattern in which the column occurrences appear. For example, related column occurrences of a Call Itemization List would be the Date, Time, Location, Phone Line Called, Number of Minutes and Charge for a particular call. You would expect these data to appear on the same row, therefore you must define the columns to extract these data on the same row.
- By default, the order in which you create table columns becomes the order in which they appear to the user online. You can move your columns left or right in any order.

Creating an Extraction Rule (Overview)

To create an extraction rule:

1. Click on the appropriate tool icon for the type of extraction rule you want to create.
2. In the DefTool work area, locate the data string you are building your rule upon, and while left-clicking your mouse, drag and drop your pointer over the start of the data string, and release the mouse-click. A ‘marquis-style’ black box appears over this selection in the work area. This is the basis for your window coordinate set, the properties of which you may edit later.
3. A Properties or Information dialog appears over DefTool for the particular type of extraction rule you are creating.
4. Name your extraction rule (if applicable). Keep names short but descriptive, as in TotAmtDue. Field, table, group, and marker names must start with a letter, use alphanumeric characters only (no special characters), and cannot contain spaces. Do not use the following reserved words in a name: Time, Date, INT, Double, Currency (“Date” is not permitted but “DueDate: is). Field, table

and group names (but not marker names) must be unique. Maximum length is 30 characters. Names are not case-sensitive.

5. Specify the data pattern to search for.
6. Create additional properties for your rule, if applicable (formatting specifications, marker options, etc.)
7. Test your rule by clicking the **Test** button (this performs a quick check on the current statement page of your DefTool work area by highlighting in purple what could be found with your rule settings).
8. Save your rule by clicking **OK** to close the Properties or Information dialog. DefTool color-highlights the area indicated by your window coordinate set. The color indicates the type of rule you created:

   Red - Document style and additional page styles

   Blue - Fields

   Bright Green - Field anchors, page style anchors and marker anchors

   Yellow - Start and end markers

   Aqua - Table columns

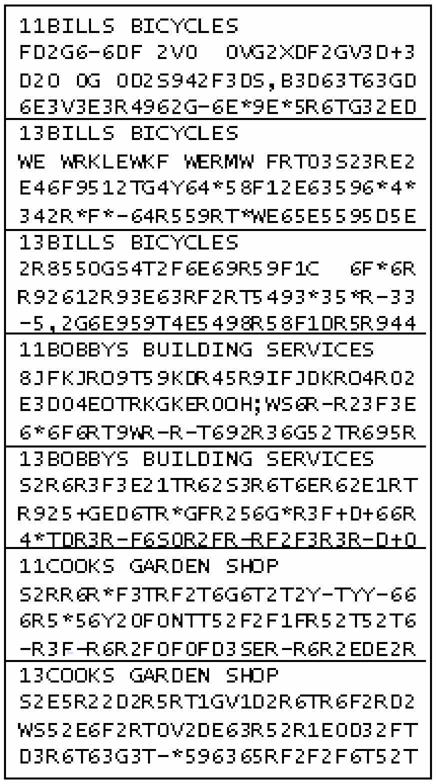
Document Style (Start of a New Document)

Every DDF requires a document style definition, indicating where each customer document starts in the data source. You must create the document style definition **before** creating a primary key or other defining other elements in the DDF.

You define the document style by specifying a name and unique pattern that the eStatement Manager Indexer production job can use to identify, upon locating this unique pattern, that the information that follows is the first page of a new statement.

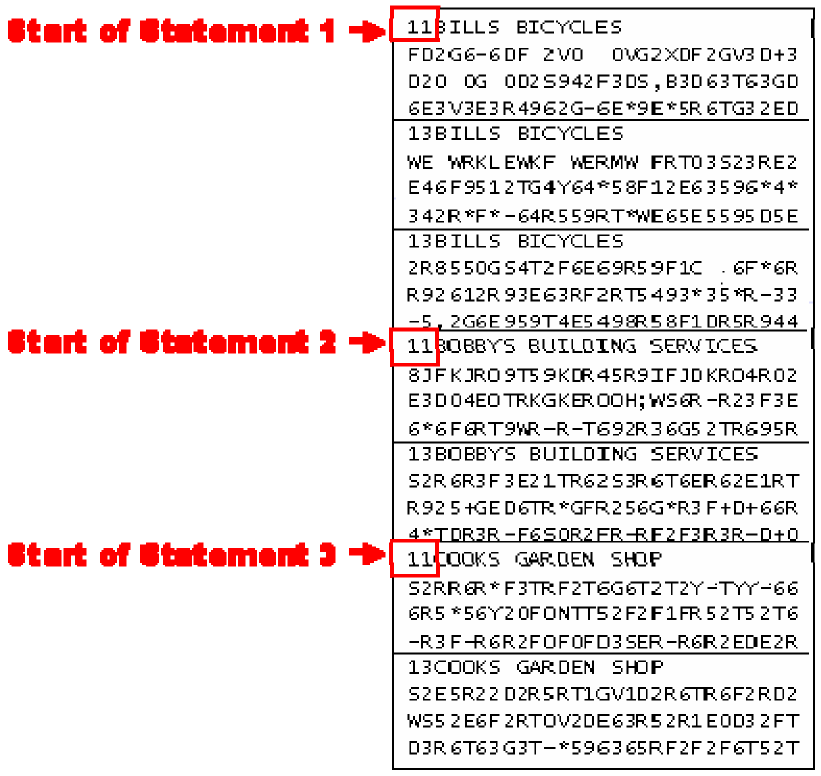
Indexer considers all the pages from the start of document until the next occurrence of the pattern to be part of one statement. You can define document style directly or after defining a document style anchor.

In the following example of a data source, each section, denoted by a line, represents a different physical page of data:



```
11BILLS BICYCLES
FD2G6-6DF 2VO  OVG2XDF2GV3D+3
D2O OG OD2S942F3DS,B3D63T63GD
6E3V3E3R4962G-6E*9E*5R6TG32ED
13BILLS BICYCLES
WE WRKLEWKF WERMW FRTO3S23RE2
E46F9512TG4Y64*58F12E63596*4*
342R*F*-64R559RT*WE65E5595D5E
13BILLS BICYCLES
2R855OGS4T2F6E69R59F1C  6F*6R
R92612R93E63RF2RT5493*35*R-33
-5,2G6E959T4E5498R58F1DR5R944
11BOBBYS BUILDING SERVICES
8JFKJRO9T59KDR45R9IFJDKRO4RO2
E3DO4EOTRKGKEROOH;WS6R-R23F3E
6*6F6RT9WR-R-T692R36G52TR695R
13BOBBYS BUILDING SERVICES
S2R6R3F3E21TR62S3R6T6ER62E1RT
R925+GED6TR*GFR256G*R3F+D+66R
4*TDR3R-F6SOR2FR-RF2F3R3R-D+O
11COOKS GARDEN SHOP
S2RR6R*F3TRF2T6G6T2T2Y-TYY-66
6R5*56Y2OFONTT52F2F1FR52T52T6
-R3F-R6R2FOFOFD3SER-R6R2EDE2R
13COOKS GARDEN SHOP
S2E5R22D2R5RT1GV1D2R6TR6F2RD2
WS52E6F2RTOV2DE63R52R1EOD32FT
D3R6T63G3T-*596365RF2F2F6T52T
```

The following image shows how DefTool uses a document style defined as "11" to identify the beginning of a new statement:

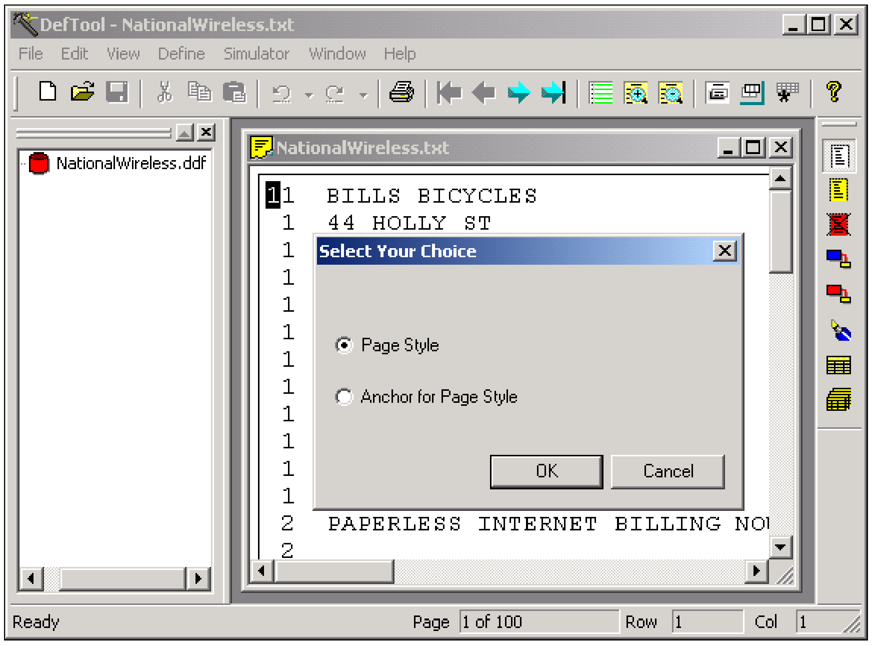


```
11BILLS BICYCLES
FD2G6-6DF 2VO  OVG2XDF2GV3D+3
D2O OG OD2S942F3DS,B3D63T63GD
6E3V3E3R4962G-6E*9E*5R6TG32ED
13BILLS BICYCLES
WE WRKLEWKF WERMW FRT03S23RE2
E46F9512TG4Y64*58F12E63596*4*
342R*F*-64R559RT*WE65E5595D5E
13BILLS BICYCLES
2R8550GS4T2F6E69R59F1C .6F*6R
R92612R93E63RF2RT5493*35*R-33
-5,2G6E959T4E5498R58F1DR5R944
11BOBBYS BUILDING SERVICES
8JFKJRO9T59KDR45R9IFJDKRO4R02
E3D04EOTRKGKEROOH;WS6R-R23F3E
6*6F6RT9WR-R-T692R36G52TR695R
13BOBBYS BUILDING SERVICES
S2R6R3F3E21TR62S3R6T6ER62E1RT
R925+GED6TR*GFR256G*R3F+D+66R
4*TDR3R-F6SOR2FR-RF2F3R3R-D+0
11COOKS GARDEN SHOP
S2RR6R*F3TRF2T6G6T2T2Y-TYY-66
6R5*56Y20FONTT52F2F1FR52T52T6
-R3F-R6R2FOFOFD3SER-R6R2EDE2R
13COOKS GARDEN SHOP
S2E5R22D2R5RT1GV1D2R6TR6F2RD2
WS52E6F2RTOV2DE63R52R1EOD32FT
D3R6T63G3T-*596365RF2F2F6T52T
```

CAUTION: Anchors provide a second level of guarantee that the information you have specified is exact. If an anchor is defined, eStatement Manager looks on each page of the file for the anchor pattern. If eStatement Manager finds the anchor, it searches for the document style. If it does not find the anchor, the DefTool does not search for the document style on that page, and therefore does not consider that page to be part of a new document. You can create a document style anchor with the same tool as for the actual document style, however anchors for document style should not be used unless absolutely necessary.

To define the document style:

1. Select the Document Style icon.
2. In the DefTool's Work Area, locate the data string that will identify the start of statement, and then click and drag your mouse over the first position of the unique pattern that signifies the start of document, and release your mouse. (Select the first **1** in **11**.) The Select Your Choice dialog opens, presenting two options: **`Page Style`** or **`Anchor for Page Style`**.
3. Accept the default of **`Page Style`**, since you can be sure that 11 will not appear in this location on subsequent statement pages. The Page Style Information dialog opens.



```
11  BILLS BICYCLES
 1  44 HOLLY ST
 1
 1
 1
 1
 1
 1
 1
 1
 1
 1
 2  PAPERLESS INTERNET BILLING NO
 2
```

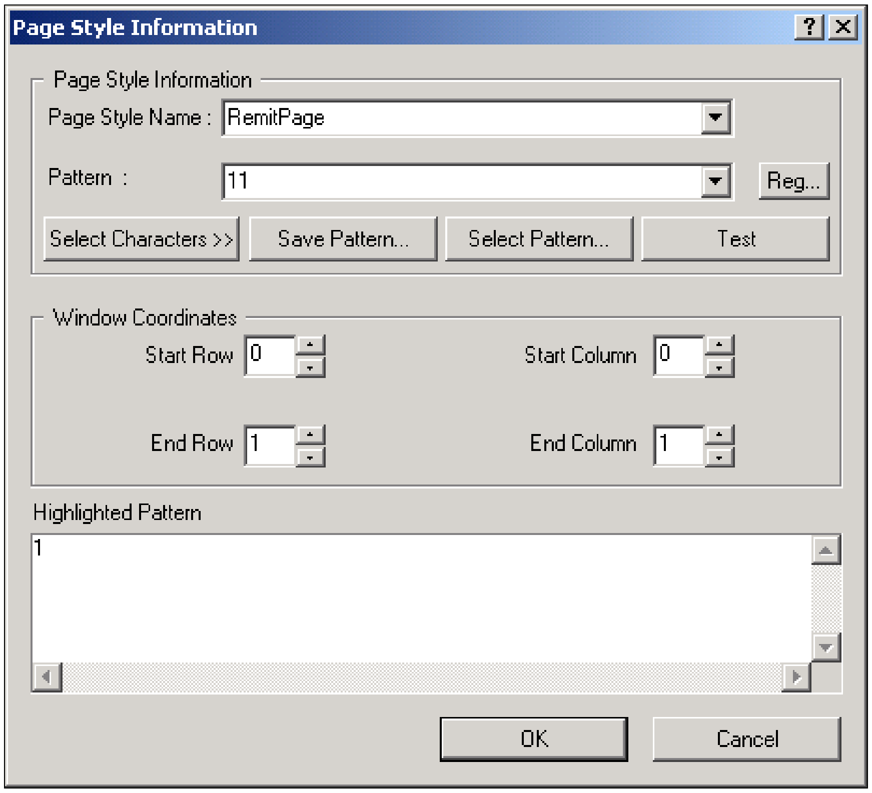
(●) Page Style
( ) Anchor for Page Style

TIP: The window area you select will define the area that is searched for the **start of the data string** on all statement pages across the data source. You may need to adjust the resulting window coordinates if you find that this data string's start location can vary. You do not need to set your window coordinates to encapsulate the entire string.

Again, the window coordinates define for the extraction components where to search for the start (or position #1) of the data string you are trying to find. The number 11 is selected as the pattern for the document style because this pattern of characters appears on the first page of all statements in the data source. Thus, it can be used as a unique identifier for the first page of a document.

You would create an anchor for a page style when the data string you are using to identify your new page style may also appear on other page styles. You could, in this case, locate an additional data string that 1) appears with (and only with) the page style string and 2) appears in the same relative location as the page style string. There can be more than one document style pages in a statement. The location of a document style AND a primary key identifies the page as the first page of the statement.

Say, for example, that the string 11 could also appear on subsequent statement pages, but that on the 'first' pages, there also existed a string called **REMIT TO:.** You could create a page style anchor that first searched for **REMIT TO:** and, if found, searched for 11. It wouldn't matter if the string **REMIT TO:** appears on subsequent pages. It is the joint appearance of **REMIT TO:** and 11 that verifies the document (or first) page style.



4 Enter your page style end. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length. (Enter **`RemitPage`**.)

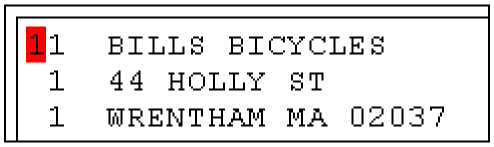
5 Enter the pattern for searching. (Enter **`11`**.)

Note that if you have selected the entire identifying data string in the work area, you can copy **`(Crtl-C)`** and paste **`(Crtl-V)`** the pattern in the Highlighted Pattern window.

6 If necessary, adjust the window coordinates (start and end rows; start and end columns) so that your coordinate set is sufficiently large to capture the start of all occurrences of your page style string, across all statements. Remember: the whole string DOES NOT need to be in the window. (The window coordinate set for the RemitPage properties should be:)

Start Row = 0	Start Column = 0
End Row = 1	End Column = 1

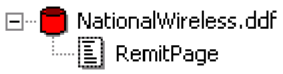
(If the first 1 in 11 is highlighted as shown below, your coordinates will match those above.)



```
11  BILLS BICYCLES
 1  44 HOLLY ST
 1  WRENTHAM MA 02037
```

Data sources often experience “float,” where data shifts position across the statements. Sometimes, this is as little as one character left or right. In other cases, data can span many rows up or down. You will need to familiarize yourself with your data source and take any “float” up/down or left/right into account when defining your window coordinates. To do this, you may have to navigate through several statements to determine what window coordinates will suffice to locate the start of your page style string. Since the DefTool main console is still active when the Page Style dialog is open, you can easily navigate to other 'first' pages with the blue arrow tools. When you need to adjust your window coordinates, you can use the up/down arrows for each corresponding coordinate field, in the Properties' Window Coordinates section. The live DefTool work area will be immediately updated with your new coordinates, by auto adjusting the original graphic selection that you made.

7 Click **OK**. The Page Style Information dialog closes, and the area you defined as the window coordinate set for the page style appears in red in the DefTool work area (shown below). A white document icon now appears in the Application Tree, representing your document style definition. (A white document icon named **RemitPage** appears in the Tree View, under the DDF parent tree node as shown below.)



CAUTION: The red highlighted area visually indicates the window coordinate set in which the DefTool will look for the start of the pattern that is identified. In the example above, the data string 11, the defined pattern, can only start in one position, between rows 0-1 and columns 0-1. If the string appeared between other rows or columns, the extraction components would not be able to locate it. Ensure that the window coordinate set is defined on a sufficient number of rows and columns to locate the desired string.

Document Recipient (Primary Key)

After you create the document style, you must define a primary key for your DDF. The primary key defines how to uniquely identify each statement contained in the data source, typically each recipient across a statement cycle.

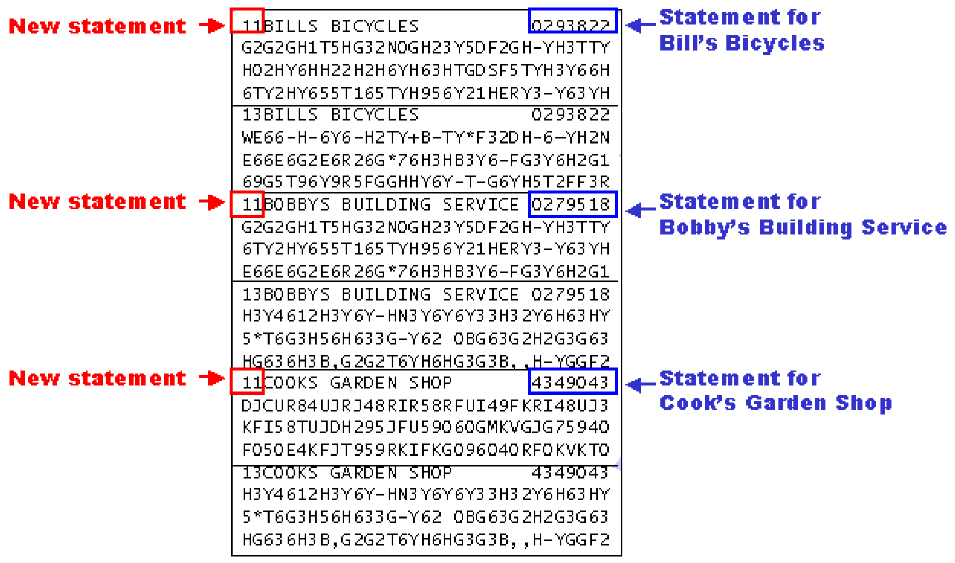
The primary key field must be unique and reside on the document style page on every statement.

CAUTION: You must define the primary key AFTER defining the document style. You must also create the primary key before creating the rest of the rules or saving your new DDF.

The account number, if the statements describe only one account, may be a good choice for the primary key field. The customer name would be a poor choice, as it is possible for two people to share the same name.

The primary key field is best defined directly, without an anchor. It is technically possible to use a field anchor with the primary key field, but is not recommended, given the importance of finding an occurrence of this field on every statement.

In the following example, the red selections indicate the data string used to define the document style. The blue selections show where the primary key field is defined. Here, the 7-digit account number is the primary key:


New statement
New statement
New statement
Statement for Bill's Bicycles
Statement for Bobby's Building Service
Statement for Cook's Garden Shop

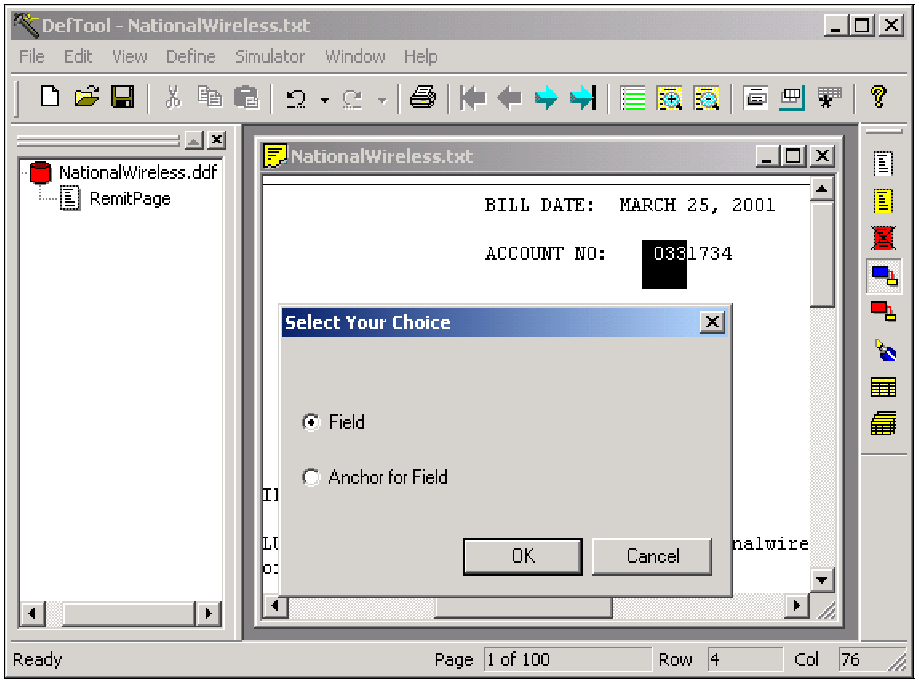
```
11BILLS BICYCLES                0293822
G2G2GH1T5HG32NOGH23Y5DF2GH-YH3TTY
HO2HY6HH22H2H6YH63HTGDSF5TYH3Y66H
6TY2HY655T165TYH956Y21HERY3-Y63YH
13BILLS BICYCLES                0293822
WE66-H-6Y6-H2TY+B-TY*F32DH-6-YH2N
E66E6G2E6R26G*76H3HB3Y6-FG3Y6H2G1
69G5T96Y9R5FGGHHY6Y-T-G6YH5T2FF3R
11BOBBYS BUILDING SERVICE 0279518
G2G2GH1T5HG32NOGH23Y5DF2GH-YH3TTY
6TY2HY655T165TYH956Y21HERY3-Y63YH
E66E6G2E6R26G*76H3HB3Y6-FG3Y6H2G1
13BOBBYS BUILDING SERVICE 0279518
H3Y4612H3Y6Y-HN3Y6Y6Y33H32Y6H63HY
5*T6G3H56H633G-Y62 0BG63G2H2G3G63
HG636H3B,G2G2T6YH6HG3G3B,,H-YGGF2
11COOKS GARDEN SHOP             4349043
DJCUR84UJRJ48RIR58RFUI49FKRI48UJ3
KFI58TUJDH295JFU59060GMKVGJG75940
F050E4KFJT959RKIFKG096040RFOKVKTO
13COOKS GARDEN SHOP             4349043
H3Y4612H3Y6Y-HN3Y6Y6Y33H32Y6H63HY
5*T6G3H56H633G-Y62 0BG63G2H2G3G63
HG636H3B,G2G2T6YH6HG3G3B,,H-YGGF2
```

To define the primary key field:

1 From the Definition Tool Bar, select the Field icon.

2 In the work area, on a first page of any statement, scroll to the location of the data string that you have chosen as the primary key field, so that you can see the entire string in the work area. You can use any statement in the data source as the basis for the creation of your primary key field definition. In fact, this is true for any extraction rule you create. Just choose the correct page style (one that contains the data desired) so that you can be accurate in your window coordinates. (Scroll to the Account Number data string, in the upper right corner of the page.)

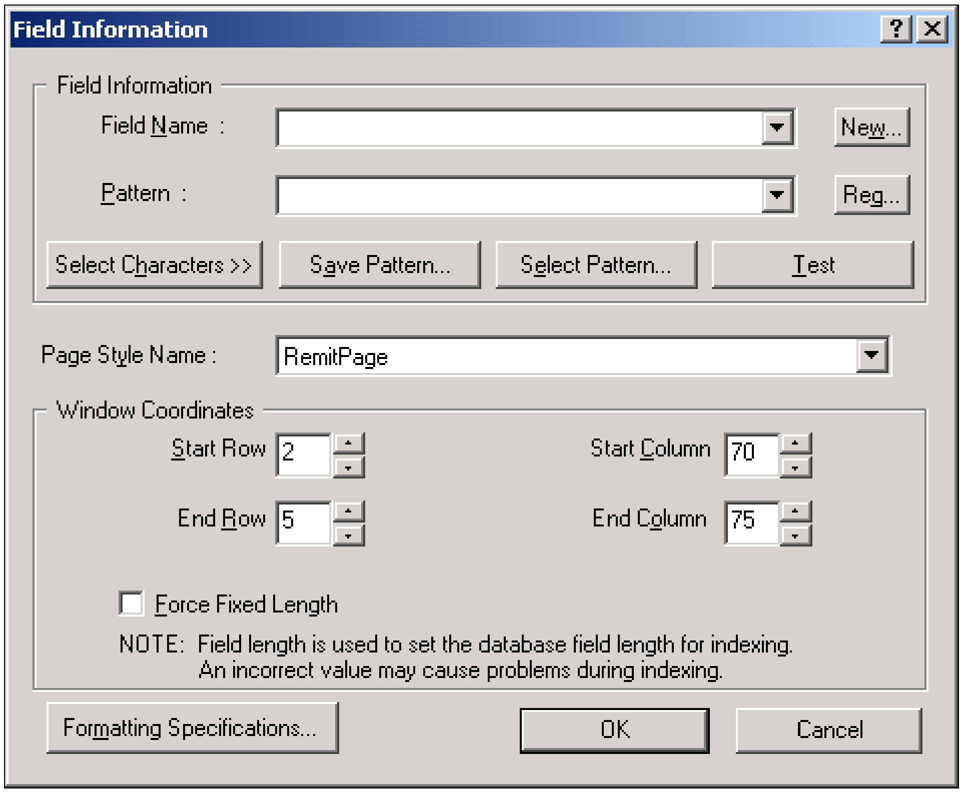
3 Use your mouse to left-click and drag over the area that contains at least the first position of the data string that you want. Select an area that includes the first numeric digit of the Account Number located to the right of the "Account Number" header. Customer names or addresses should not be used as the primary key. Social Security numbers, account numbers, or any other unique item is recommended. The area is highlighted in black, and the Select Your Choice dialog box appears, displaying two options: Field or Anchor for Field.



```
BILL DATE:  MARCH 25, 2001

ACCOUNT NO:  0331734
```

Select Your Choice: (●) Field ( ) Anchor for Field — OK, Cancel

4 Select **Field** and click **OK**. The field Information dialog opens.



Field Information

Field Name :

Pattern :

Page Style Name : RemitPage

Window Coordinates

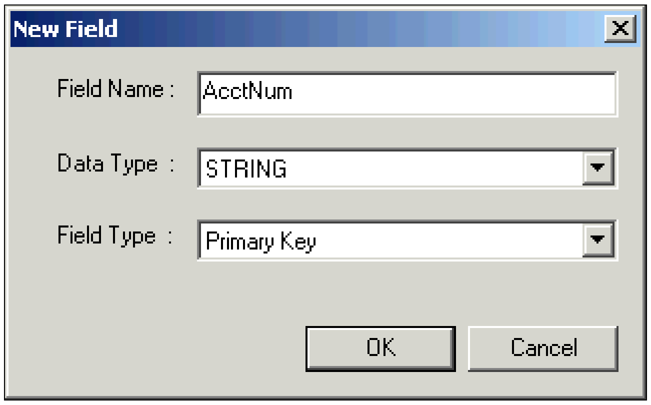
Start Row 2 Start Column 70

End Row 5 End Column 75

[ ] Force Fixed Length

NOTE: Field length is used to set the database field length for indexing. An incorrect value may cause problems during indexing.

5 In the Field Information dialog, select the **New** button. The New Field dialog opens.



Field Name : AcctNum

Data Type : STRING

Field Type : Primary Key

6 Enter Primary Key **`Field Name`**. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length. (Enter **`AcctNum`**.)

7 Select **`STRING`** from the Data Type dropdown list.

CAUTION: Your **`Primary Key field`** must be defined as a **`STRING.`**

8 From the Field Type dropdown box, select **`Primary Key`**. The only time you will select a field type is during the Primary Key field definition.

9 Click **`OK`**. The New Field dialog closes and the name appears in Field Name.

10 Enter the pattern to extract the primary key. (Enter the regular expression **`[0-9]+`**)

11 Select the page style name from the drop down window. (Select **`RemitPage`**.)

12 Verify the window coordinates. Your window coordinates must include the first position of the Account Number data string. You should navigate through other 'first pages' to verify that these coordinates are sufficient to capture the data string on all statements. If you find that the window needs adjustment, you can make changes within the Field Information dialog, in the Window Coordinates section, by clicking the up/down arrows for each coordinate.

After navigating through the work area view of your statements' first pages, determine the appropriate window coordinate set. An appropriate window coordinate set would be: Start Row = 2, Start Column = 72, End Row = 4, End Column = 76

Since this definition is for the Primary Key field, you need to define a fixed length of characters for the extraction components. Use information from your Oracle Bill Mastering sessions (and also from the hard-copy samples of your statements) to determine what this length should be.

CAUTION: Using an anchor for the Primary Key field is not recommended unless necessary.

13 Determine the maximum length of characters for the Account Number data string by either navigating through the hard copy or soft-copy of the National Wireless statement sample.

14 Enter the fixed length desired into the **`Length`** field, and check on the **`Force Fixed Length`** option. This setting will force the extraction components to verify the first part of the data string found, by the pattern, and then extract any subsequent characters to the number specified in the Length field. (Enter a length of **`7`** characters, and select the **`Force Fixed Length`** option.)

15 Click **`OK`** to close the Field Information dialog. The Field Information window closes. Your Primary Key field name now appears in Application Tree, as a child node of the Document Style tree node.

Creating Additional Page Styles

A page style is a rule you create to define the style for pages subsequent to the document style (first) page in a statement.

Your statements will likely have additional pages that contain different types of data and have an appearance different from the document style. For example, your statements may require additional page styles to include an Account Summary, Detail, Cover Letter, ID Card or other information.

Also, the same statement can have one or more pages of the same page style, such as a single account summary page followed by 25 detail pages on a credit card statement. You must create two page styles for a DDF to extract data from a statement like this. The first page style would be the account summary page. If it is also the first page of the statements, then this would be the document style. The second page style would be the detail page.

Creating an additional page style is nearly the same process as defining a document style. You must find a data string on the page (i.e. the title DETAIL) that uniquely identifies this page style, and define the area and pattern to allow the extraction components to locate this string during deployment.

You can define a page style in two ways:

- Directly
- After the definition of a page style anchor

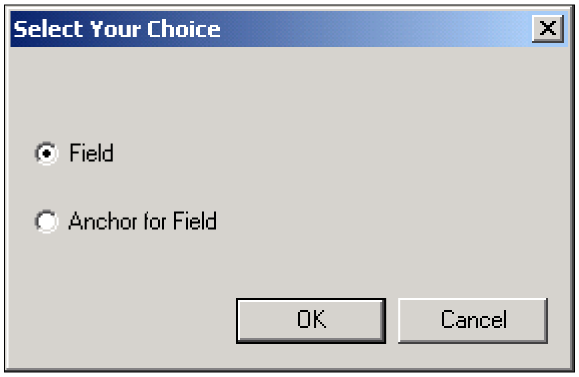
If you've already defined an anchor, the system looks for the anchor first and then the page style when extracting data. The directions below do not address defining a page style anchor. The process to create a page style anchor does not differ from the creation of a field anchor, with the exception that you must use the page style tool. Thus, refer to the Field Anchors Overview for more detail on anchors.

CAUTION: You must define the rule to detect your page styles before you build rules to extract data from these pages. You cannot create a field, for example, to be extracted from a Detail page, until you have created a page style to properly locate the Detail pages of a statement.

To determine what page styles will need to be defined, refer to the information that you acquired during your Oracle Bill Mastering sessions. (Refer to the hard-copy views of your National Wireless statements.)

To define a page style:

1. Navigate to the first statement's second page of the data source by clicking the Next Page icon. (Go to Page 2 of the *NatlWireless.txt* data source in the DefTool work area.)
2. Click the Page Style icon.
3. In the work area, left-click and drag your mouse over the first position of the data string that will identify the new page style. (In the work area, select the **A** in **`ACCOUNT SUMMARY`**.) The Select Your Choice dialog appears (shown below).



4 Select **`Page Style`** or **`Anchor for Page Style`**. (Select **`Page Style`**.)

5 Click **`OK`**. The Page Style Information dialog opens.

6 Enter the page style end. (Enter **`AcctSummary.`**)

CAUTION: Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length.

7 Enter a regular expression (static or variable) to validate the identifying string. (Enter **`ACCOUNT SUMMARY`**.)

8 Verify that your window coordinates are sufficient to capture the first position of your identifying string, on all pages of this type, across all statements in your data sources. (Verify that your AcctSummary page style window coordinates are: Start Row = 2, Start Column = 33, End Row = 3, End Column = 34)

9 Click **`OK`** to close the Page Style Information dialog and save your page style. (Click **`OK`**. Your new page style, **`AcctSummary`**, is now represented in the Application Tree, and you can now build rules to extract data from the Account Summary statement pages.)

Overview of Fields and Field Anchors

Once you have defined the document style and primary key rules, you can write extraction rules for data fields.

Before you can begin writing field extraction rules, you must carefully analyze the input file to:

- Identify all the pieces of information you need to extract from the data input file to show customers in online statements, such as account number, address, services, total amount due, and statement date.
- Identify any data you want to extract for use with conditional business logic. For example, you may want to display different-looking statements for B2B or B2C customers. By extracting the customer type data field, you can write a conditional statement that looks at the field value to determine which HTML template to use for each customer. (You build conditional logic into your application using Composer.)

Note that you should hard-code static data into the HTML template instead extracting it from every statement. This includes labels or other descriptive text such as the "**`ACCOUNT NUMBER:"`** or "DUE BY:" labels that accompany the variable account number and due by fields every customer has. Occasionally, you may need to extract descriptive, static text along with the variable data. This is usually the case when the data does not always exist in every statement, such as Long Distance Charges. This way, you can drive the placement of descriptive text by the presence of relevant data.

You can define a field on its own or after the definition of a field anchor.

You can also define any post-extraction formatting rules you want.

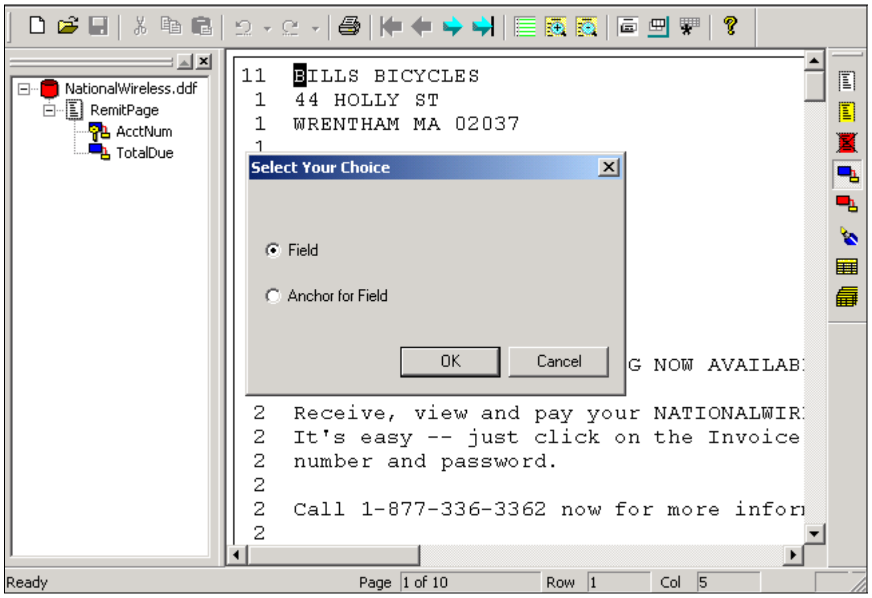
Creating a Field Definition

This section describes how to create a field definition.

CAUTION: Be sure to include all fields you need to index when you create the indexing DDF. Although you can add new fields to a DDF later, you must manually add the new columns to your database index table.

To create a field definition:

1. In the DefTool work area, navigate to any occurrence of the page style that contains your prospective field, and scroll to the location of the field. (Navigate to the **`RemitPage`** of any statement, i.e. Page 1 of the data source. Scroll to the upper left corner of the page, where the customer name resides.)
2. If you are creating a field directly, ensure that the **`New Field`** icon is selected. If you have created an anchor for your new field, this icon will already be selected.
3. In the DefTool work area, left-click and drag your mouse over the area that covers the first position of all customer names in the statements of the data source. The Select Your Choice dialog appears. (Left-click and drag your mouse over the first character in the customer name, i.e. highlight the **`B`** in **`BILLS BICYCLES.`**)



```
11  BILLS BICYCLES
 1  44 HOLLY ST
 1  WRENTHAM MA 02037
 1
```

Select Your Choice

[✓ Field]

Anchor for Field

OK   Cancel

```
                              G NOW AVAILAB
 2  Receive, view and pay your NATIONALWIR
 2  It's easy -- just click on the Invoice
 2  number and password.
 2
 2  Call 1-877-336-3362 now for more infor
 2
```

Ready   Page 1 of 10   Row 1   Col 5

4. Select **`Field`** and click **`OK`**. The Field Information dialog appears.
5. Select the **`New`** button. The New Field dialog opens.

6 Enter a name and data type for your field. Keep names short but descriptive, as in TotAmtDue. Z_DocDate, Z_PRIMARY, Z_DOC_ID, and Z_CONTEXT are reserved eStatement Manager field names. Note that if you do not define a Z_DocDate document date field, eStatement Manager automatically uses the date indexed as the document date. eStatement Manager uses this field as the basis for displaying the statement summary, or hit list. Field, table, group, and marker names must start with a letter, use alphanumeric characters only (no special characters), and cannot contain spaces. Do not use the following reserved words in a name: Time, Date, INT, Double, Currency ("Date" is not permitted but "DueDate:" is). Field, table and group names (but not marker names) must be unique. Maximum length is 30 characters. Names are not case-sensitive. (Enter **`CustomerName`** as the field name, and accept the default of **`STRING`** as the data type.)

CAUTION: **`Do not`** change Field Type (this box remains empty unless defining primary key field.

7 Click **`OK`**. The New Field dialog closes and the name appears in the Field Name box.

8 Click **`OK`** to close the New Field dialog.

9 In the Pattern box, enter an appropriate regular expression that will be used to validate and extract the field data. (Enter the regular expression **`[A-Z]`** to find only strings that start with a capital letter.)

At this point, you should decide if your field extraction rule will use a pattern to extract the entire data string, or if you will extract the field by Forced Fixed Length. If you want to extract the field by length, you may still enter a pattern that will validate at least the first portion of the data value. For example, to avoid extracting lines of spaces, you should use " **^** " (not a space) as your pattern. This will require that the first position contain a character other than a space.

10 Select the page style where the field resides from the corresponding dropdown list. (Select **`RemitPage`**.)

11 Verify that the window coordinates are sufficient to capture the first position of any occurrence of your field, across all statements. (Your window coordinates should be: Start Row = 0, Start Column = 4, End Row = 2, End Column = 5.)

6 Custom Post-Conversions

Using Custom Post-Conversions

You can create custom post-conversions to use with the data extraction rules you define using DefTool. DefTool lets you apply post-conversion formatting specifications to a field or column value to further format the data for dynamic display online or for data storage purposes using the Command Center Indexer job. You can currently apply standard positional-based alphanumeric and date post-extraction conversions to extracted data, such as "Delete all trailing zeros or spaces™" and "Translate Aug 21, 2002 to 08/21/2002™." You can create *customized* post-conversions to apply along with any sequence of existing (standard) post-conversions. You can apply multiple standard or custom post-conversions to a data field.

You can create custom post-conversions using:

- Regular expressions using the third-party PCRE 3.9 RegEx library for non-date and non-time data
- Date and time string formats

New regular expression library

When creating a regular expression for custom post-conversions in eStatement Manager, you must use the syntax of the third-party PCRE 3.9 RegEx library. For information about this library and the use of regular expressions, see "Mastering Regular Expressions" by Jeffrey E.F. Friedl, available at http://www.oreilly.com/catalog/regex/toc.html

It is important to note that is not the same library you use to create regular expressions for extracting data using the DDF.

Date and time formats

See "Appendix A – Data Definitions" for a list of date and time formats you can use to build post-conversions for date and time data in eStatement Manager.

Reusing custom post-conversions

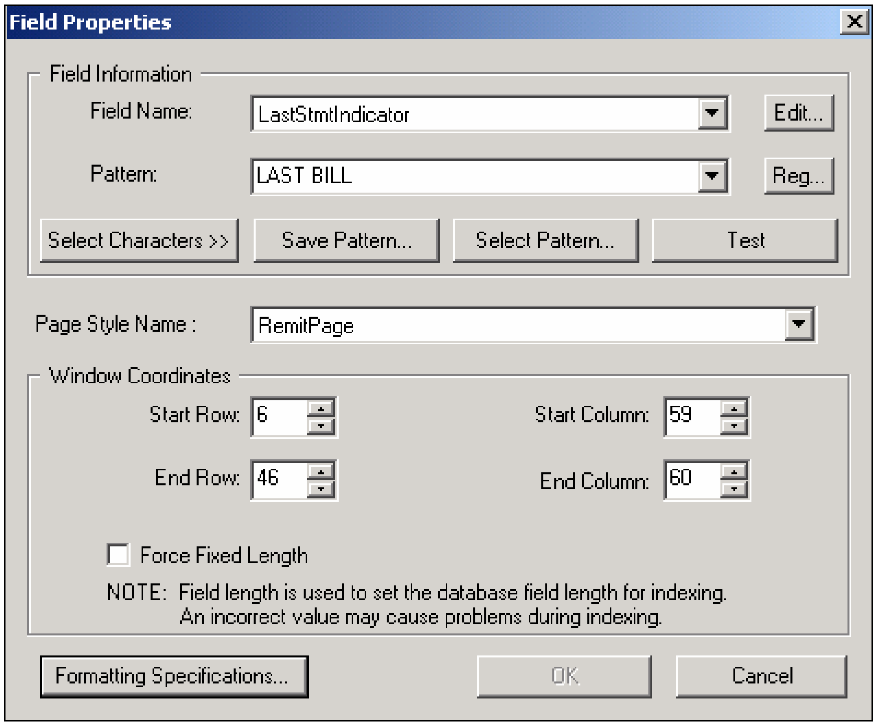
Each custom post-conversion you create becomes available for reuse with other data fields in the same eStatement Manager application (in the same DDF).

In addition, DefTool lets you save an application's custom post-conversions to an xml file. This enables you to individually copy and paste the expressions into custom post-conversions you create in other another DDF.

Creating a New Custom Post-Conversion

To create a custom post-conversion for a field or column:

1. When defining the extraction rules for a field, DefTool displays the Field Properties dialog. (When defining a column, DefTool uses the similar Column Properties dialog.)



Field Properties

Field Information

Field Name: LastStmtIndicator

Pattern: LAST BILL

Page Style Name : RemitPage

Window Coordinates

Start Row: 6

End Row: 46

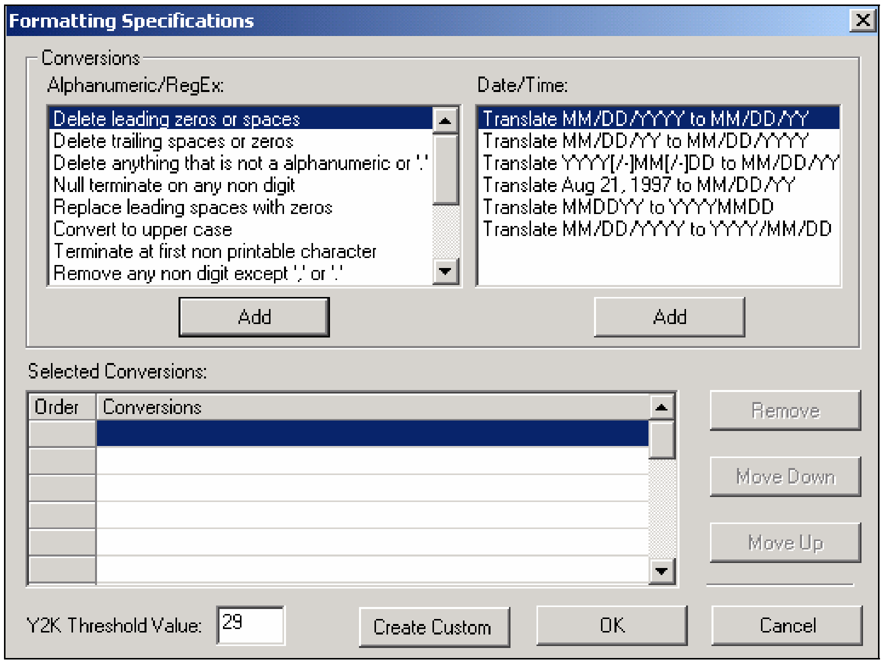
Start Column: 59

End Column: 60

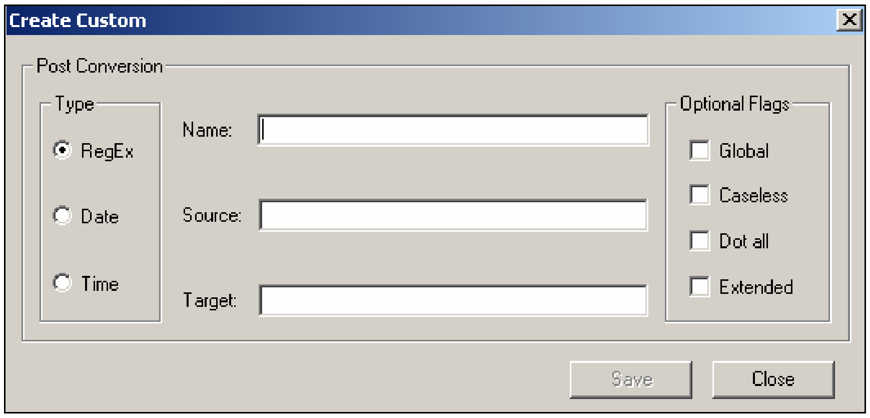
☐ Force Fixed Length

NOTE: Field length is used to set the database field length for indexing. An incorrect value may cause problems during indexing.

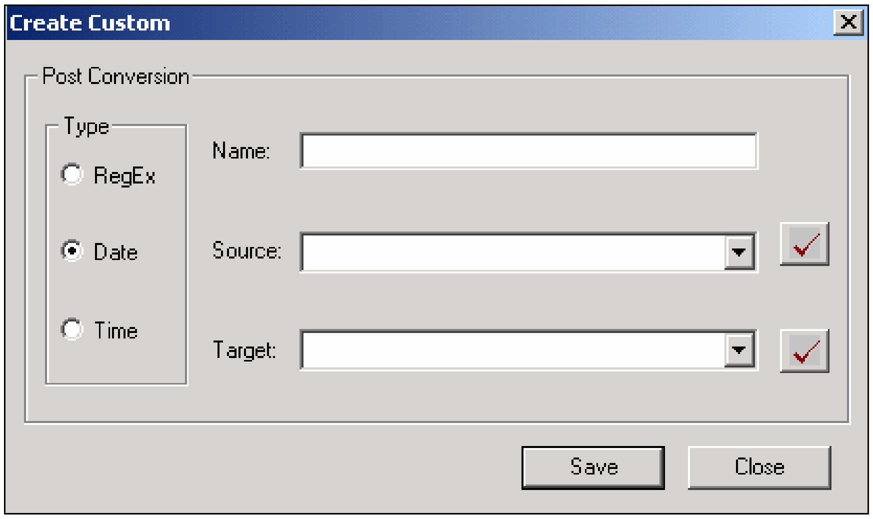
2. Click **`Formatting Specifications`**. DefTool displays the Formatting Specifications dialog. Use this dialog to choose any combination of standard or custom post-conversions and their application sequence to apply to the field or column.


Formatting Specifications
Conversions
Alphanumeric/RegEx:
Delete leading zeros or spaces
Delete trailing spaces or zeros
Delete anything that is not a alphanumeric or '.'
Null terminate on any non digit
Replace leading spaces with zeros
Convert to upper case
Terminate at first non printable character
Remove any non digit except ',' or '.'
Date/Time:
Translate MM/DD/YYYY to MM/DD/YY
Translate MM/DD/YY to MM/DD/YYYY
Translate YYYY[/-]MM[/-]DD to MM/DD/YY
Translate Aug 21, 1997 to MM/DD/YY
Translate MMDDYY to YYYYMMDD
Translate MM/DD/YYYY to YYYY/MM/DD
Add
Add
Selected Conversions:
Order
Conversions
Remove
Move Down
Move Up
Y2K Threshold Value: 29
Create Custom
OK
Cancel

3 Click **Create Custom**. DefTool displays the Create Custom dialog.



4 Specify the type of post-conversion you want to create: Date or Time to convert most date or time data, or RegEx to specify a regular expression for all other data types. If you choose a date or time, the Create Custom dialog redisplays with different options.



Type: RegEx [✓ Date] Time

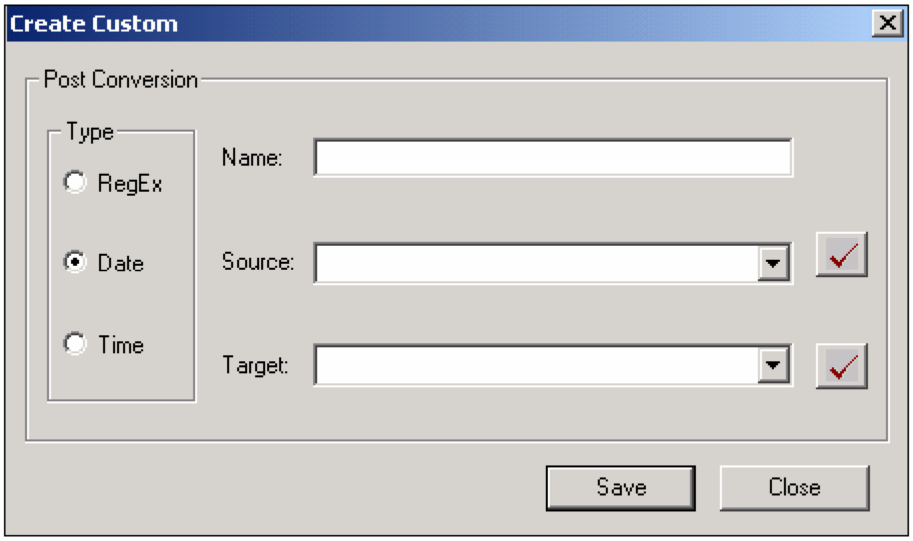
5 Specify a meaningful name to identify the custom post-conversion. (This name appears in the "Alphanumeric/RegEx" or "Date/Time" conversion list on the Formatting Specifications dialog after you save, making it available to select with other fields or columns.)

6 In the Source field, enter the regular expression or date/time format functions you want to apply to the extracted data (this is on top of the regular expression defined on the Field Properties dialog that eStatement Manager first applies to the extracted data). For a regular expression, you can choose any of the following matching option letters; DefTool adds them to the expression between "(?" and ")".

- g - Global
- i - Caseless
- s - Dotall
- x - Extended

7 In the Target field, enter the expression for converting the extracted source data for presentment.

8 When creating a date or time post-conversion, click ✓ after specifying the source and target functions to verify that the format is valid.



Type: RegEx [✓ Date] Time

9 When you're through specifying the post-conversion functions, click **Save**. Continue defining any additional post-conversions you need, then click **`Close`**. DefTool displays the Formatting Specifications dialog with the new custom post-conversion in the appropriate conversion list.

10 To add the custom post-conversion to the field, highlight the conversion and click **Add**. Click **`OK`** when you are done defining formatting specifications for this field.

11 You can continue the process of defining extraction rules for this field.

Editing a Post-Conversion

To edit a custom post-conversion:

1 With the DDF open in DefTool, select **`View>Custom List`**. The Custom Post Conversion List dialog appears.

2 Highlight the custom post-conversion you want and click **`Edit`**. The Edit Custom dialog appears.

3 Enter your edits. Click **Save**.

4 Click **`Close`**.

Removing a Custom Post-Conversion from a Field or Column

To delete a post-conversion (custom or standard) from the formatting specification for a particular field:

1 With the DDF open in DefTool, right-click the field or column in the tree.

2 Select **`Edit`** from the right-click menu.

3 Click **Formatting Specifications**.

4 Highlight the custom post-conversion in the Selected Conversions list and click **Remove**.

5 Click **OK**.

6 Click **OK** on the Field (or Column) Properties dialog.

Deleting a Custom Post-Conversion from an Application

To delete a custom post-conversion from the application:

1 With the DDF open in DefTool, select **View>Custom List**. The Custom Post-Conversion List dialog appears.

2 Highlight the custom post-conversion you want and click **Delete**.

3 If the selected custom post-conversion is not in use, DefTool asks if you are sure you want to delete it. If it is in use, DefTool tells you which fields currently use the custom post-conversion and warns you that deleting it also removes it from the field formats. Click **Yes** to delete the custom post-conversion (or **No** to cancel). DefTool removes the custom post-conversion from the application and from any fields where it was in use.

4 Click **Close**.

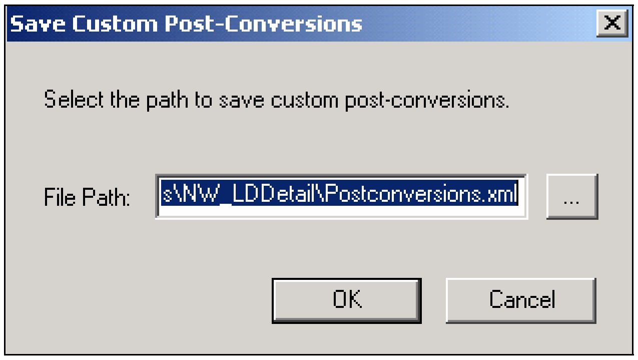
Sharing Custom Post-Conversions with other Applications

When you create and save a custom post-conversion, it becomes available for use in that application DDF only. To make those custom post-conversions available for use in another DDF, DefTool lets you save the custom post-conversions to a text file (in xml format). You must then recreate the custom post-conversions in the other DDF, copying and pasting the individual expressions from the file.

You can also add custom post-conversions to an existing file.

To save custom post-conversions to an external text file:

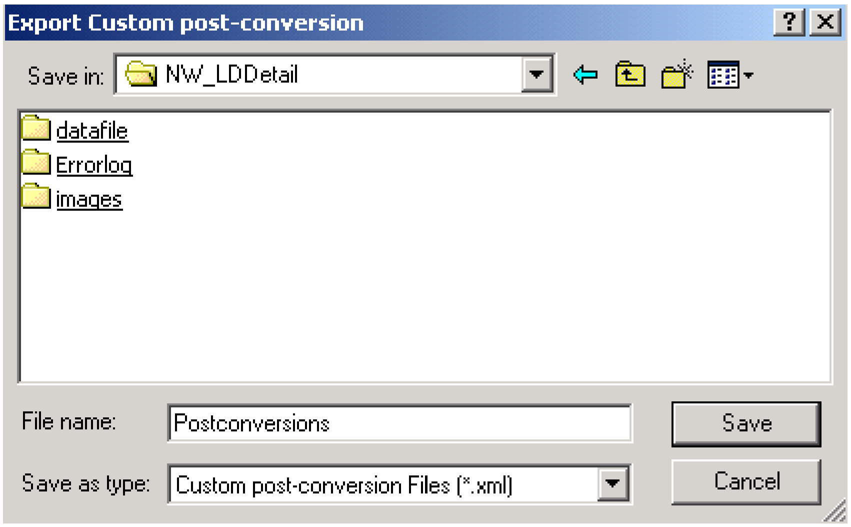
1 **Select File>Save custom post-conversions**. The Save Custom Post-Conversions dialog appears.


Save Custom Post-Conversions

Select the path to save custom post-conversions.

File Path: s\NW_LDDetail\Postconversions.xml

2 Click ... to browse for a directory and/or edit the file name. DefTool displays the Export Custom Post-Conversion dialog. Specify the path and/or file name and click **Save**.


Export Custom post-conversion

Save in: NW_LDDetail

datafile
Errorlog
images

File name: Postconversions

Save as type: Custom post-conversion Files (*.xml)

3 Click **OK**.

To copy saved custom post-conversions to another DDF:

1 In DefTool, open the DDF you want to add the saved custom post-conversions to.

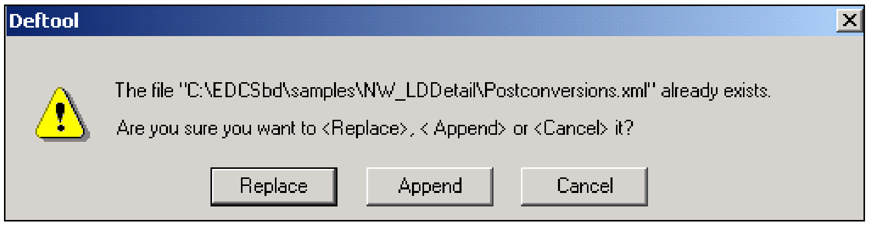
2 In a text editor, open the xml file containing the saved custom post-conversions.

3 Follow the instructions in this chapter to create custom post-conversions, copying and pasting the individual expressions from the xml file to recreate the custom post-conversions in this DDF.

To append or replace an existing custom-post-conversions file:

1 Select **`File>Save Custom Post-Conversions`**. The Save Custom Post-Conversions dialog appears.

2 Click ... to browse for the xml file you want to append or replace. DefTool displays the Export Custom Post-Conversion dialog:


Deftool

The file "C:\EDCSbd\samples\NW_LDDetail\Postconversions.xml" already exists.

Are you sure you want to <Replace>, < Append> or <Cancel> it?

3 Click **Append** or **`Replace`**. DefTool adds any new custom post-conversions to the file or replaces the file contents entirely.

Date and Time Format Constraints

See “Appendix A: Data Definitions” for details about the basic date and time formats available.

You must observe the following restrictions when using date and time formats to create custom post-conversions in DefTool:

- When converting a time duration format to another duration format, use only the following string components: %s, %Q, %K and %G
- You cannot convert a time format to a time duration or vice-versa. For example, you cannot convert the time duration “%K:%Q” to time format “%H:%M:%S” or the reverse. (It is OK to convert a time format to another time format, or to convert a time duration to a time duration, however.)
- Do not include both time duration formats and time formats in either the source or target. For example, you cannot have a source format of “%H:%Q:%S” and make the target “%T” because “%Q” represents a duration and “%H” is a time format.
- You cannot convert minutes to hours. Since the clock is in 1000 minutes, you can specify only %Q for the source. This only stores minutes in the tm structure and will contain values up to 999 and does not populate the hour field.

Examples of Custom Post-Conversions

The following examples show the regular expressions and formats you would use in the Source and Target fields on the Create Custom dialog to create various custom post-conversions.

Post-Conversion	**Source (Regular Expression)**	**Target (Format**	**Flag (Mode)**
1,234,567.89 to **1.234.567,89** Convert dot(.) to comma(,) and comma(,) to dot(.)	`(\.){1}|(\,)`	`(?1,)(?2.)`	`/g` (Global)
1234567.89 to **1.234.567,89** Insert thousand separator as dot(.) and decimal separator as comma(,)	`(\.){1}|\G(\d{`	`(?1,)(?2$&.)`	`/g` (Global)
00001234567.89 to **1.234.567,89** Remove leading zeros; insert thousand separator as dot(.); change decimal separator to comma(,)	`(\.){1}|^0+|\G(\d{1,3})(?=(?:\d\d\d)+(?!\d))`	`(?1,)(?2)(?3$&.)`	`/g` (Global)
-1234567.89 to **-1.234.567,89** Maintain negative sign; insert thousand separator as dot(.); change decimal separator to comma(,)	`(\.){1}|\G\-?(\d{1,3})(?=(?:\d\d\d)+(?!\d))`	`(?1,)(?2$&.)`	`/g` (Global)
-1,234,567.89 to **-1.234.567,89** Maintain negative sign; change thousand separator to dot(.) and decimal separator to comma(,)	`(\.){1}|(,)`	`(?1,)(?2.)`	`/g` (Global)
€1,234,567.89 to **€1.234.567,89** Maintain the currency sign; change decimal separator to comma(,) and thousand separator to dot(.)	`(\.){1}|(,)`	`(?1,)(?2.)`	`/g` (Global)
€-78912354354.65767 to **€-78.912.354.354,65767** Maintain currency sign; change decimal separator to comma(,); insert thousand separator as dot(.)	`(\.){1}|\G(€?|\-?|€?\-?)(\d{1,3})(?=(?:\d\d\d)+(?!\d))`	`(?1,)(?2$&.)`	`/g` (Global)
€-0000587365456 56.74647467 to **€-58.736.545.656,74647467** Maintain currency sign; change decimal separator to comma(,); insert thousand separator as dot(.); trim off leading zeros	`€?(0+)|€\-(0+)|\-?(0+)|(\.){1}|\G(€?|\-?|€?\-?)(\d{1,3})(?=(?:\d\d\d)+(?!\d))`	`(?1€)(?2€-)(?3-)(?4,)(?5$&.)`	`/g` (Global)

Post-Conversion	**Source (Regular Expression)**	**Target (Format**	**Flag (Mode)**
00000000.234 to **0,234** Trim off leading zeros; change decimal separator to comma(,)	`(0){1,}(\.)`	`0,`	`/g` (Global)
000000001.234 to **1,234** Trim off leading zeros; change decimal separator to comma(,)	`(\.)|0+(\.){1}|^(0)+`	`(?1,)(?20,)(?3)`	`/g` (Global)

9999999999 to **(999)999-9999**
This requires you create and apply two separate, sequential post-conversions.
	Post-conversion 1:
	`\G^(\d{1})|\G(\d{5})`
	Post-conversion 2:
	`\G(\()+(\d{3})`
		Post-conversion 1:
		`(?1\($&)(?2$&-)`
		Post-conversion 2:
		`(?1$&\))`	

10/14/2002 to **October 14, 2002**	%m/%d/%Y	%B %d, %Y	

18 julio 2002 or **20 agosto 2002** or **05 mayo 2001** to
18-07-2002 or **20-08-2002**
Use these conversions for dates in languages other than English and French.
This requires you create and apply two separate, sequential post-conversions.
	Post-conversion 1 (Regular expression):
	`(?:enero)|(?:febrero)|(?:marzo)|(?:abril)|(?:mayo)|(?:junio)|(?:julio)|(?:agosto)|(?:septiembre)|(?:octubre)|(?:noviembre)|(?:diciembre)`
	Post-conversion 2 (Date conversion):
	%d %m %Y
		Post-conversion 1 (Regular expression):
		`(?101)|(?202)|(?303)|(?404)|(?505)|(?606)|(?707)|(?808)|(?909)|(?1010)|(?1111)|(?1212)`
		Post-conversion 2 (Date conversion):
		%d-%m-%Y
			`/g` (Global)

Explanation of first example

The first example in the table above shows that to convert 1,234,567.89 to 1.234.567,89, the post-conversion you specify must change the dot(.) to a comma(,) and the comma(,) to a dot(.).

Source – The regular expression looks for exactly one occurrence of dot(.) or a comma(,) in the source data:

```
(\\.){1}|(\\,)
```

Target – The target format specifies that if condition 1 is satisfied, replace the dot with a comma; else if condition 2 is satisfied, replace the comma with a dot:

```
(?1,)(?2.)
```

Flag (mode) – The `/g` or Global flag applies the formatting information globally (throughout the input string).

Explanation of second example

The second example in the table above shows that to convert 1234567.89 to 1.234.567,89, the post-conversion you specify must search for and insert the thousand separators as dot(.) and the decimal separator as comma(,).

Source – The regular expression looks for exactly one occurrence of a dot(.) or sets the anchor to the beginning, if a digit between exactly one to three occurrences, look ahead three digits followed by not a digit:

```
(\.){1}|\G(\d{1,3})(?=(?:\d\d\d)+(?!\d))
```

Target – The target format specifies that if condition 1 is satisfied, then replace the dot with a comma, else if condition 2 is satisfied, then insert a dot(.):

```
(?1,)(?2$&.)
```

Flag (mode) – The `/g` or Global flag applies the formatting information globally (throughout the input string).

7 About Field Anchors

What are Field Anchors?

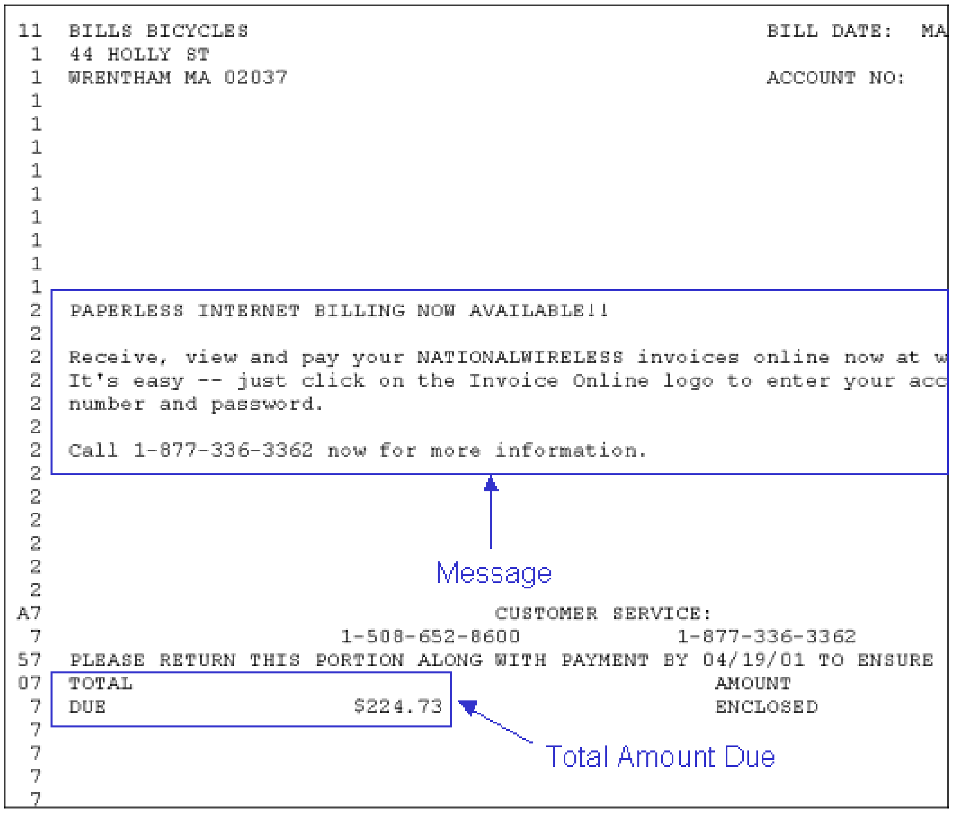
Field anchors provide a verification that the data that is found during extraction of a field is, in fact, the correct data. If a field anchor is defined, the extraction components look in the defined anchor window coordinates for the anchor pattern. If the anchor pattern is found, the extraction components then search for the field. If the anchor is not found, the DefTool does not search for the field on that page, and thus, does not extract any field data for that statement.

Field anchors are commonly used when the field data can “float” widely from statement to statement. If the window coordinate set of a field can span many rows, for example, then there is the strong possibility that other data strings of similar pattern could appear in the defined location. This means that the field rule you write could potentially extract the wrong data string. In this case, you would locate a data string that:

- Appears only if the field data exists on the statement
- Appears in a location relative to the field data.

Field anchors are commonly used with currency fields, especially when many currency values are on a page. Consider the following example:

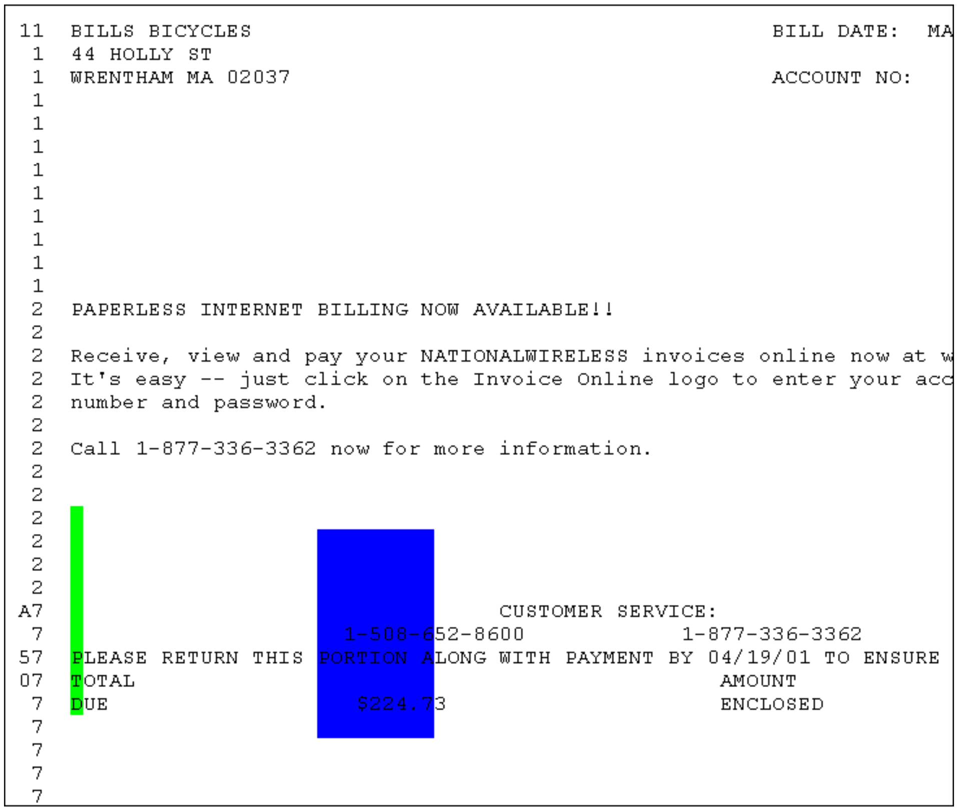
On a biller’s statement, the Total Amount Due data resides on the Remittance Page. The Remittance Page (defined as the document style because it is the first page of the statement), may include a message to the recipient, the length of which can vary (illustration shown below).



```
11  BILLS BICYCLES                                              BILL DATE:  MA
 1  44 HOLLY ST
 1  WRENTHAM MA 02037                                           ACCOUNT NO:
 1
 1
 1
 1
 1
 1
 1
 1
 1
 2  PAPERLESS INTERNET BILLING NOW AVAILABLE!!
 2
 2  Receive, view and pay your NATIONALWIRELESS invoices online now at w
 2  It's easy -- just click on the Invoice Online logo to enter your acc
 2  number and password.
 2
 2  Call 1-877-336-3362 now for more information.
 2
 2
 2
 2
 2
 2
A7                                    CUSTOMER SERVICE:
 7                      1-508-652-8600              1-877-336-3362
57  PLEASE RETURN THIS PORTION ALONG WITH PAYMENT BY 04/19/01 TO ENSURE
07  TOTAL                                                  AMOUNT
 7  DUE                        $224.73                     ENCLOSED
 7
 7
 7
 7
```

The Message here spans 7 rows, however, there is no guarantee that it will always span 7 rows on other statements. If the Message is shorter, then the Current Charges and Total Due values' will shift upwards. Additionally, the Message may not exist on some statements.

In-depth analysis of your data is needed to determine how widely the location of a data string can vary. So if the Message can be up to 10 lines or not exist at all (0 lines), then the row span that is necessary to extract the Total Due field is 10 rows.



```
11  BILLS BICYCLES                                          BILL DATE:  MA
 1  44 HOLLY ST
 1  WRENTHAM MA 02037                                       ACCOUNT NO:
 1
 1
 1
 1
 1
 1
 1
 1
 1
 2  PAPERLESS INTERNET BILLING NOW AVAILABLE!!
 2
 2  Receive, view and pay your NATIONALWIRELESS invoices online now at w
 2  It's easy -- just click on the Invoice Online logo to enter your acc
 2  number and password.
 2
 2  Call 1-877-336-3362 now for more information.
 2
 2
 2
 2
 2
 2
A7                                    CUSTOMER SERVICE:
 7                      1-508-652-8600             1-877-336-3362
57  PLEASE RETURN THIS PORTION ALONG WITH PAYMENT BY 04/19/01 TO ENSURE
07  TOTAL                                               AMOUNT
 7  DUE                      $224.73                    ENCLOSED
 7
 7
 7
 7
```

TIP: The data movement dictates the column span of a window coordinate set as well. Remember that your window coordinate set is used to locate the first position of the data string you are extracting. In the case of currency values, which tend to be right justified, you should compensate for large and small values, and adjust your column span to detect them (shown above).

The TotalDue field can contain the following definition:

```
Start Row/Column = 21, 4
End Row/Column = 30, 5
Pattern = $[0-9,]*\.[0-9][0-9][0-9][CR]*
```

Remember that the extraction components search the area defined in a left-to-right, top-to-bottom order. By defining a rule to locate another data string first, before searching for Total Due, we can safely extract the correct value.

When creating a field anchor, you must choose a data string that occurs in the same location relative to the data. The anchor string you choose can be on the same row as the data, or on the rows above or below, as long as the anchor string is always in that location relative to the data on every statement.

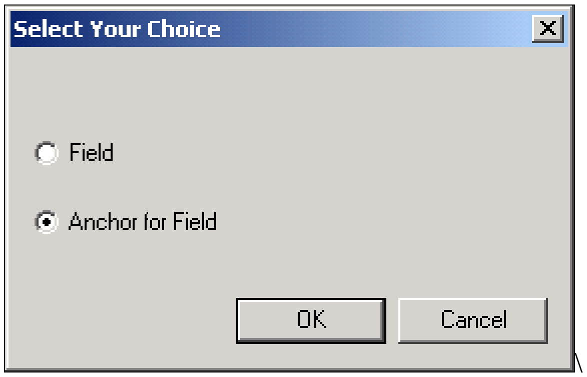
In the example above, after Bill Mastering, we can say that the string TOTAL always occurs on the same row as the TotalDue field value, therefore it would make a good anchor string.

Creating a Field Anchor

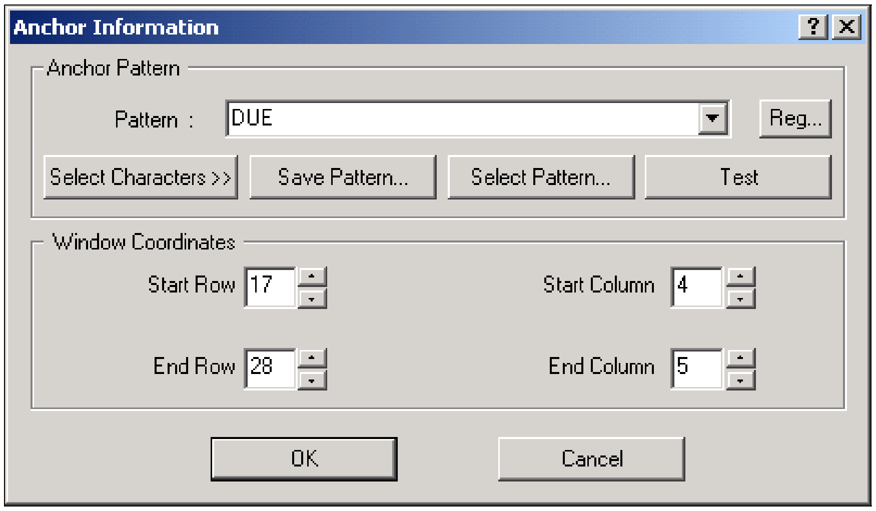
You create field anchors using the same tool as for field creation. When you need to create an anchor for a field, you must create it immediately before you create the field. This two-step process ensures that an internal bond is created between the anchor and the field.

To create a field anchor:

1. From the Definition Tool Bar, click the New Field icon.
2. In the DefTool work area, select the area that you expect the anchor string to start on any statement. In this case, you would highlight rows 17 to 28, since you expect this string to be aligned with the Total Due value, and columns 4 to 5, since the string is left-justified and does not vary column-wise. The Select Your Choice dialog appears.
3. Click the **`Anchor for Field`** radio button (as shown) and click **OK**. The Anchor Information dialog appears (shown below).



In the Pattern field, enter the static string or variable regular expression that will detect your anchor string. In this case, you would enter **DUE**.



Pattern : DUE

Start Row 17 Start Column 4

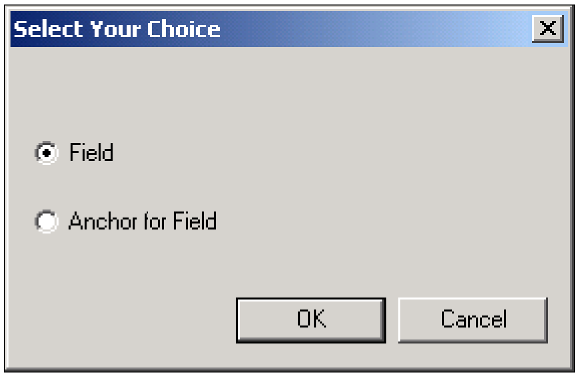
End Row 28 End Column 5

You can test this definition on the work area view by clicking the Test button. This will highlight, in purple, any string that this definition would be able to pick up.

4 Click **OK** to save changes and close the Anchor Properties dialog.

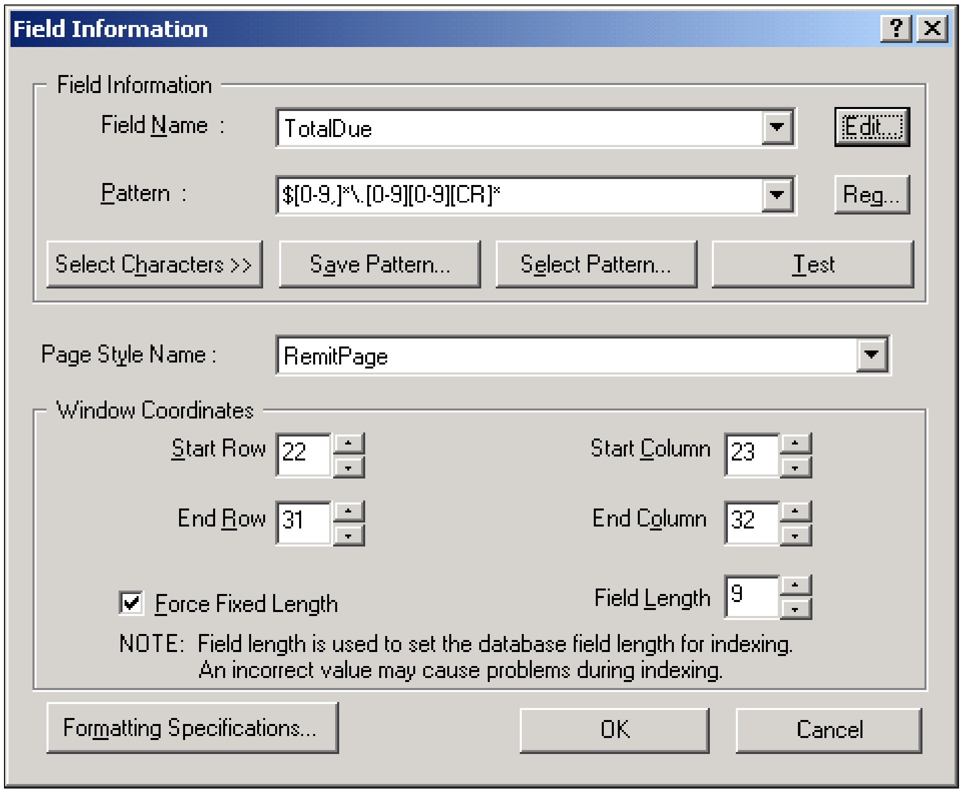
CAUTION: In the DefTool work area, you will notice that the area defined by the anchor's window coordinate set is now highlighted in bright green. There is no Application Tree item yet; the anchor definition will be contained within the field that you are about to create. This is why **you must create the field immediately after the anchor.**

5 With the New Field icon still selected in the Definition Tool Bar, highlight the work area to locate the field's first position. The Select Your Choice dialog appears again.


Select Your Choice

(● Field) ○ Anchor for Field

6 This time, specify Field (as shown) and click **OK**. The Field Information dialog appears (shown below).


Field Information

Field Information
Field Name : TotalDue
Pattern : $[0-9,]*\.[0-9][0-9][CR]*

Page Style Name : RemitPage

Window Coordinates
Start Row 22
Start Column 23
End Row 31
End Column 32
[✓] Force Fixed Length
Field Length 9

NOTE: Field length is used to set the database field length for indexing. An incorrect value may cause problems during indexing.

7 Name the field in Field Name. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length.

8 Select the data type of the field.

9 Select the page style where the field resides in the statement.

10 Specify the pattern and/or set a length to extract by.

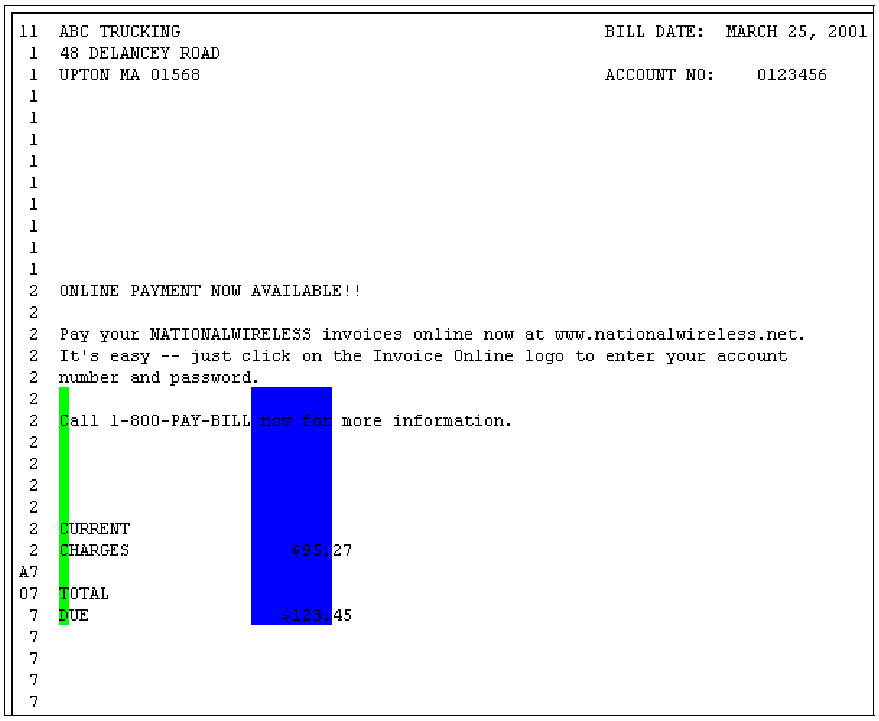
11 Make other formatting specifications as desired.

12 Adjust your window coordinates if necessary.

13 Click **OK** to save and close the Field Information dialog.

The anchor definition is bound internally to the field that is defined immediately after the anchor, and there is no Tree View representation of the anchor. You cannot delete an anchor without deleting the field to which it is attached.

The DefTool Tree View gains a new node for the field you created. When clicked, the work area is highlighted to show both the field and anchor windows (shown below).



```
11  ABC TRUCKING                                        BILL DATE:  MARCH 25, 2001
 1  48 DELANCEY ROAD
 1  UPTON MA 01568                                      ACCOUNT NO:    0123456
 1
 1
 1
 1
 1
 1
 1
 1
 1
 2  ONLINE PAYMENT NOW AVAILABLE!!
 2
 2  Pay your NATIONALWIRELESS invoices online now at www.nationalwireless.net.
 2  It's easy -- just click on the Invoice Online logo to enter your account
 2  number and password.
 2
 2  Call 1-800-PAY-BILL now for more information.
 2
 2
 2
 2
 2  CURRENT
 2  CHARGES                 $95.27
A7
07  TOTAL
 7  DUE                    $123.45
 7
 7
 7
 7
```

CAUTION: Notice that the windows for the anchor and field shown above have the same number of rows. It is important that the row spans of the anchor and Field windows are the same. It is also important that the placement of the anchor window mimics the positional relationship of the anchor to the field. For example, if the anchor string is expected to be two rows above the field data, then the anchor window needs to start and end two rows above the Field window.

Editing an Anchor

The anchor definition is bound internally to the field that is defined immediately after the anchor, and there is no tree representation of the anchor.

To edit an anchor:

1. Locate and highlight the anchor’s field in the DefTool Tree. As shown above, two windows will appear in the work area.
2. Locate the bright green window in the work area, and carefully double click this window. This will open the Anchor Information dialog where you originally defined the anchor.
3. Make the necessary changes to the anchor’s pattern or window coordinate set, and click **OK** to close.

Deleting an Anchor

Since the anchor is not an entity of its own, it cannot be deleted independently. To delete an anchor, you must delete the field to which it is attached, and immediately repeat the steps of creating an anchor and its field.

8 Tables

When to Use Tables

You have already learned how to identify when data should be extracted as a field. To review, you build a field rule when your data exists on one row only, and does not have multiple iterations throughout the statement.

How do you extract data that exists on multiple rows, or is related to other data items, and/or has more than one unique occurrence in a statement? When your data fits any of these criteria, you will most likely extract it in the form of a table.

Simply put, a table consists of one or more fields that are combined into a unit for extraction and presentment purposes. In addition, the fields of a table, called columns, are related to each other in some way, and can have many instances in a statement. The table functionality allows you to build rules to extract these field columns together, and look for multiple instances of the columns on all rows between a predetermined start and end location.

Extracting Tables for Text Presentation

The use of tables in customer applications will allow you to extract and present iterative or transactional information in a crisp, well-defined format that is easy to read in a browser. Some good examples of data that are best extracted as a table are:

A Customer Address — Some addresses consist of 2 lines of data, while other addresses require many more rows. The best way to extract an address entirely and simply is to create a table with one column (to extract all iterations of the Address Line)

A Credit Card Transaction List — The items of a Credit Card Transaction may include a date, retail outlet, location, and charge. Each of these items are related to each other in the transaction, therefore it makes sense to extract and present them together. Further, it is highly likely that a transaction list will contain multiple transactions, perhaps continuing the list onto many subsequent pages. If you extract this data as a table, with a predefined start and end location for the extraction components to search for data, the related items of every transaction will be presented with each other, and no special configuration is necessary.

Extracting Tables for Chart Presentation

eStatement Manager can extract data from your source, and present it dynamically in the form of a Chart. If you have the appropriate data, you can build a table to extract it and then use that table later on in the Composer, to build rules to compose the Chart during live retrieval of a statement. Charts are composed of two distinct pieces of information, namely the data to be charted and presentation directions. Using the eStatement Manager Composer tool, one can specify the data for charting. The charting functionality requires that you have KavaChart© installed.

KavaChart can load IIS or Oracle databases, and run via your own customized server-side tools. Charting in eStatement Manager involves the following steps:

1. In the Composer tool, insert a chart tag.
2. Define the look and feel properties for the chart.
3. Publish the properties file to the eStatement Manager application server.
4. Ensure display privileges are on the eStatement Manager web server.

You will need the appropriate data in your source in order to offer a Chart to your end-users. An example of Chart-appropriate data is:

- Current Service Charges — If, for example, your statements contain a Current Charge value for each service you provide, you can create a table that looks across an entire statement, with two columns to extract all instances of the following information:
- Current Charge Label — the descriptive text that occurs with the relative Current Charge (such as Local Service Charges)
- Current Charge Amount — the currency value of the Current Charge.

Essentially, you need to have data that will populate the following Chart components:

- The X-Axis, or Label
- The Y-Axis, or Value

See Chapter 5 for information about creating Charts in Composer.

Table Definitions

A table definition consists of:

- One or more columns, which are defined similarly to fields, and whose iterations occur in a pattern together.
  - A column cannot be searched for in more than one location
  - A column cannot be defined with its own anchor. Rather, it uses the table anchor by default (table anchors are described below)
- A set of markers, which, during live retrieval, dynamically define the area in a statement that will be searched for instances of the defined columns. Your options for table markers are:
  - You can define only a start marker, or a start and end marker pair
  - Markers can have multiple locations
  - A start marker and end marker do not have to exist on the same page style
- An anchor, which guarantees that, on the row that it is found, there exists relevant data to be extracted in the table. The anchor of a table can either be:

- Internal Anchor — A column that you have defined as part of the table, which will always extract data on all rows where there is table data. A good choice for an internal anchor might be the charge amount of a call itemization list, which depends on the logic that if there is a charge amount on a row between the start and end marker, then there is data on this row that should be extracted as part of the table.
- External Anchor — A data string that is not actually part of the table data but occurs on rows, between the start and end marker, where there are relevant column data. A good example of an external anchor is the use of static control codes that are not part of the table, but indicate the existence of the table data on the row in which they are located.

Creating a Table

Once you have identified the data that you want to extract as a table, create the rules that define the table:

1. Define the start markers of the table.
2. Define the end marker (optional) of the table.
3. Define the table format:
   - Name the table.

   CAUTION: Observe the following rules when creating field, table, group, and marker names:

   The names are not sensitive to case, but they must be fashioned from ONLY alphanumeric characters (no special characters are permitted).

   The names must start with a letter, and cannot contain any spaces.

   The names cannot contain any of the following reserved words alone: Time, Date, INT, Double, Currency. (For example: **`Date`** is not permitted, but **`DueDate`** is allowed.)

   Field, table and group names must be unique (marker names are exempt from this requirement).

   Keep the names short, e.g. **`TotAmtDue`**. Be descriptive, but use abbreviations instead of long names.

   - Select the marker set that will detect the table and dynamically set the searching area for the column(s).
   - Define the table column(s).
   - Define the table anchor (internal or external).

Defining a Table in National Wireless

This section will lead you through each step of creating the rules that define a table using the National Wireless application. To illustrate each step, we will define the first table of the course on the customer address.

The goal will be to extract all of the customer information in one unit, including the customer end. This will allow us to define presentation of all this information with great ease in the Composer. We already extracted the customer end as a separate field, however we are not limited to extracting it once (see the Note below).

Sometimes, you will define rules to extract the same data in different ways. Since we want to present the customer name and address together, it makes sense to extract it as a unit, or in one table. Also, you may want to index the customer name as a field in the database. Since only fields can be indexed to the database, it makes sense to also extract the customer name as a field. This "double extraction" will not have an impact on application performance.

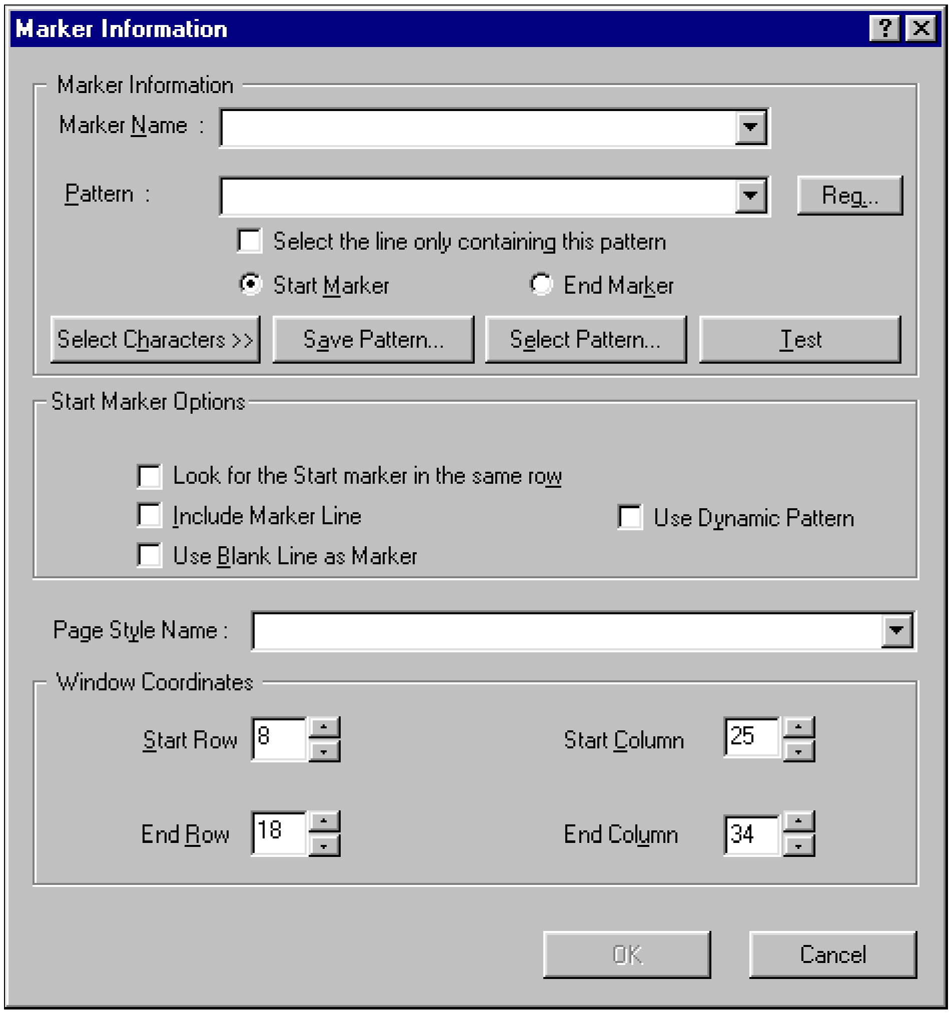
Defining a Table Start Marker

During live retrieval of a statement, a table start marker is used to inform the extraction components of the row on which to start searching for table data. You start the process of defining the start marker by first identifying a data string that appears on or above the row where table data begins. Start markers are mandatory, however, you may define more than one start marker for a table.

To define a table start marker:

1. Identify and locate the data string that indicates the start row of the table search area. This is where the extraction components will begin looking for column data. (Locate the string **11** in rows 0-1 and columns 0-1.)
2. Click to select the Marker icon from the Definition Tool Bar.

   You can easily find out the coordinates where your mouse is located. If you move your mouse into the work area, and then look at the Status Bar in the bottom pane of the screen, you can see the row and column coordinates of the cursor's location.
3. Move your mouse into the work area and left-click drag and drop your mouse over the first position of the string identified in Step **1.** (Select the 1 in 11, so that one character position is selected.) The Select Your Choice dialog opens.
4. You will need to decide if you are creating a marker or an anchor for the marker (which you would create immediately after the anchor). (Accept the default of **`marker`**, and click **`OK`**.) The Marker Information dialog appears:



Start Marker selected. Start Row 8, End Row 18, Start Column 25, End Column 34.

5 Enter the marker name. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length. (Enter **CustAddress**.)

6 Enter the pattern to be searched for as the start marker. Be sure that your pattern is sufficient to detect this start marker on all statements. (Enter **11** as the pattern.)

7 When creating a marker, the **Start Marker** radio button is selected by default. Accept this default for the start marker definition.

8 The start marker of a table can be located on or above the rows where table data exists. If your start marker string is on the same row where there may exist table data, then enable the **Include Marker Line** option. (The start marker string, 11, appears on the same row as some customer address data. Click to enable **Include Marker Line**.)

9 The start marker string can be "re-used" as the end marker, meaning that you can define the end marker on the same exact string in the same location, with the intent that it be detected on a subsequent page. If you want to use the same string for the start and end marker and **not** create a null searching area for the table data, then enable **Look for the Start Marker on the Same Row** in the Start Marker Information dialog. (For NatlWireless, since your start marker and end marker will be different for the CustAddress table, **do not** enable **Look for the Start Marker on the Same Row**.)

10 If you are certain that your start marker's row contains no other data than the start marker string itself, enable the **`Select the Line Only Containing This Pattern`** option. (Since your start marker row does contain other data, **do not** enable **`Select the Line Only Containing This Pattern`**.)

CAUTION: **`Select the Line Only Containing This Pattern`** will direct the extraction components to search the entire row for verification that no data besides the start marker string exists on that row. To use this option, you must be certain that no other data will ever appear on the start marker row.

11 You can use a row with no data on it as your start marker, by enabling the **`Use Blank Line as Marker`** option. To use this option, you must be certain that a blank row will always indicate the existence of the table, and that there will always be a blank row preceding the table. (**Do not** enable **`Use a Blank Line as marker.`**)

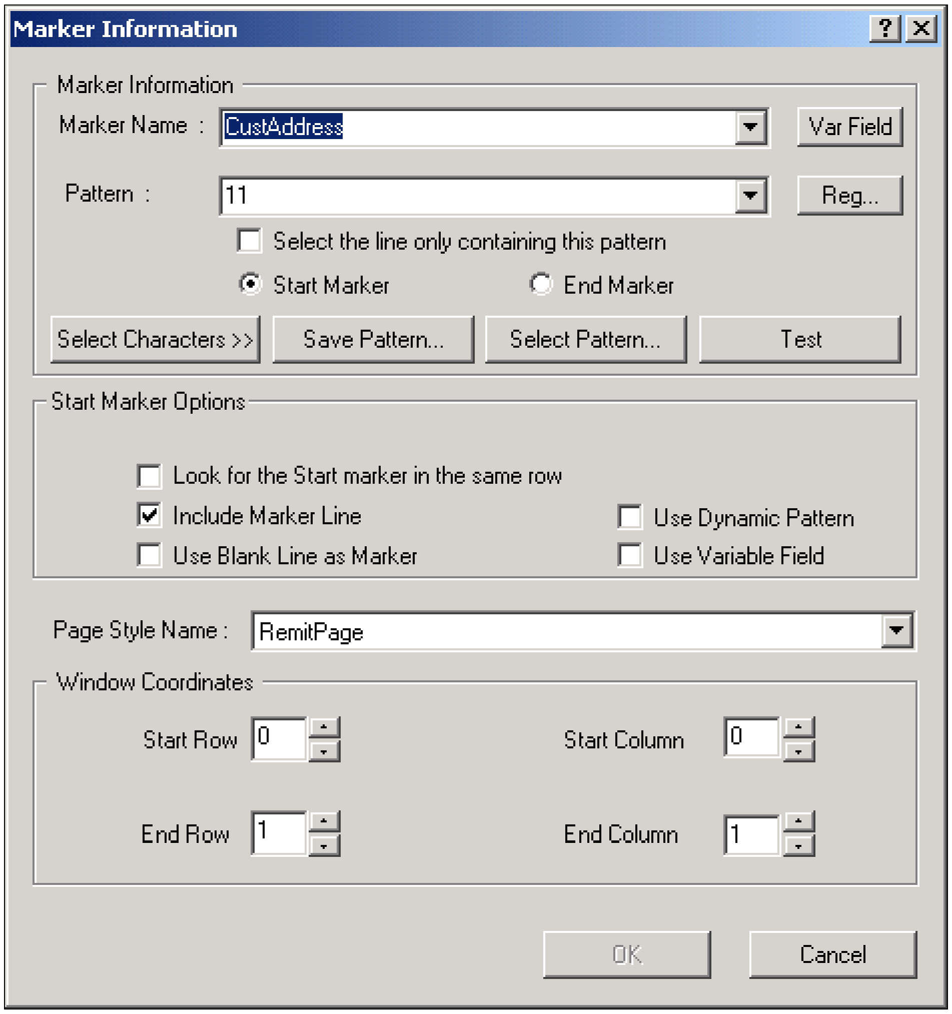
12 Select the page style on which the start marker string resides. (Select **`RemitPage`** from the Page Style dropdown list.)

13 Verify that your window coordinates are sufficient to detect your start marker on all statements. If this table will be part of a group, verify that the window coordinates, especially rows, are sufficient to detect ALL occurrences of the start marker on a statement, and across all statements. (Verify that your window coordinates are:)

Start Row = 0	Start Column = 0
End Row = 1	End Column = 1

TIP: You can create your table rules so that only sections of a table are extracted and presented in a statement, which is driven by the choice of the recipient during statement live retrieval. To do this, you use a dynamic field, which dynamically sets a pattern to a table column, therefore requiring that only data lines where column X = Y will be presented. This feature appears in the marker information dialogs to define the same type of rules to extract specific group occurrences. Since markers for groups are created the same way as markers for tables, you see the feature here, however, selective table data extraction and presentment driven by dynamic pattern matching is defined at the table column level.

Your completed Marker Information dialog appears as shown below.


Marker Information

Marker Information
Marker Name : CustAddress
Pattern : 11
[ ] Select the line only containing this pattern
[● Start Marker] ○ End Marker

Start Marker Options
[ ] Look for the Start marker in the same row
[✓] Include Marker Line
[ ] Use Blank Line as Marker
[ ] Use Dynamic Pattern
[ ] Use Variable Field

Page Style Name : RemitPage

Window Coordinates
Start Row 0
Start Column 0
End Row 1
End Column 1

14 Click **OK** to close the Marker Information dialog and save your start marker definition. In the work area, notice that a yellow area defined by your marker's window coordinate set appears over your start marker string. Also, in the Tree a Start Marker node has been added to the tree.

Defining a Table End Marker

During live retrieval of a statement, a table end marker informs the extraction components of the row on which to stop searching for table data. You start the process of defining the end marker similarly to a start marker, by first identifying a data string that appears on or below the row where table data is expected to end. Whereas at least one start marker is mandatory, however, the definition of an end marker is optional, depending on the type of data you are extracting.

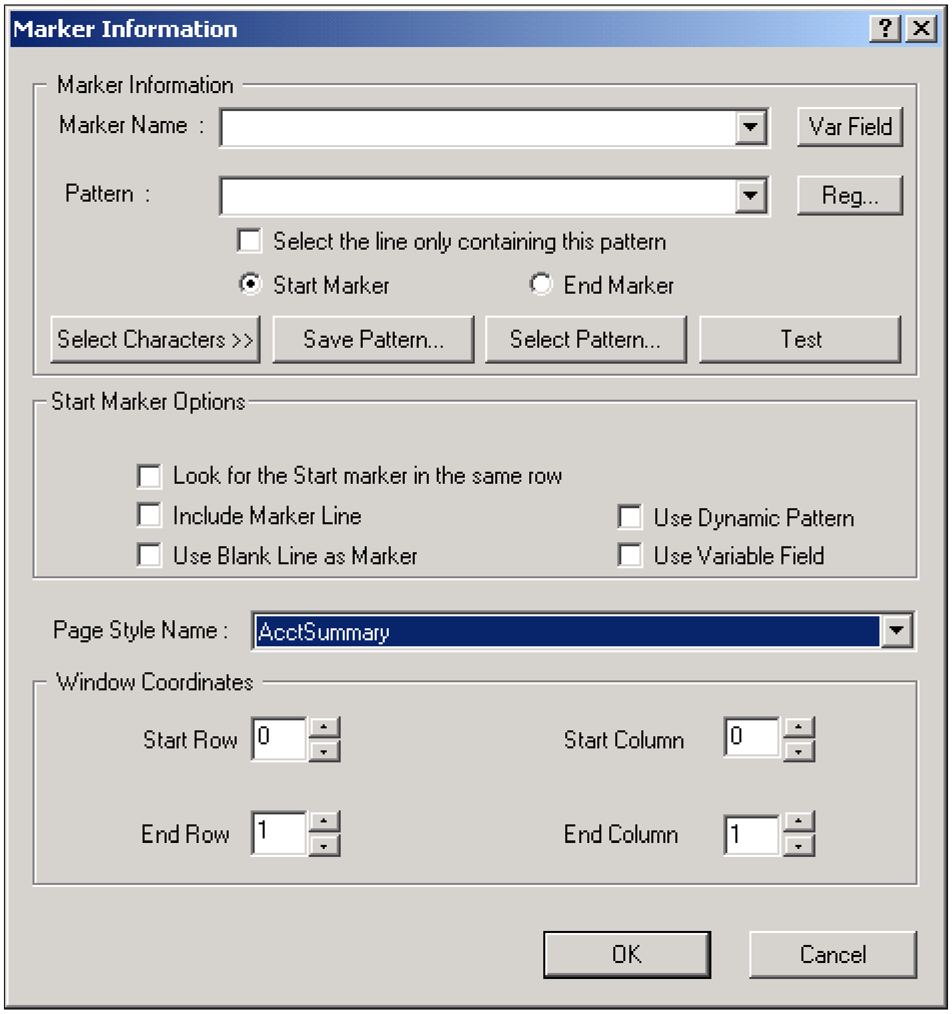
To define a table end marker:

1 Identify and locate the data string that indicates the end row of the table searching area. This is where the extraction components will stop looking for column data. (The string 2 in the left margin of the page is the best indicator of the end of the Customer Address section.)

2 Select the Marker icon from the Definition Tool Bar.

3 Move your mouse into the work area and left-click drag and drop your mouse over the first position of the string identified in Step 1. (Select the first two control characters along the left margin of the page.)

4 The Select Your Choice dialog opens. From here, decide if you are creating a marker or an anchor for the marker (which you would create immediately after the anchor). (Accept the default of **`Marker`**, and click **OK.**) The Marker Information dialog opens (shown below).


Marker Information

Marker Information

Marker Name :

Pattern :

[ ] Select the line only containing this pattern

[✓ Start Marker] End Marker

Start Marker Options

[ ] Look for the Start marker in the same row

[ ] Include Marker Line

[ ] Use Dynamic Pattern

[ ] Use Blank Line as Marker

[ ] Use Variable Field

Page Style Name : AcctSummary

Window Coordinates

Start Row 0 Start Column 0

End Row 1 End Column 1

5 Enter the marker name. Names must not contain spaces or special characters, and must start with an alpha character. Maximum length is 30 characters.

When you are creating a set of start and end markers, to use together with a table, you must give the end marker the same name as the start marker. This forces them to be attached as a marker pair. If you have already created and named the start marker to use with this new end marker, then you already have the end marker name. Select this name from the marker name dropdown list or enter it manually. (Select **`CustAddress`**.)

6 Enter the pattern to be searched for as the end marker. Be sure that your pattern is sufficient to detect this start marker on all statements. (Enter **2** as the pattern.)

7 When creating any marker, the Start Marker radio button is selected by default. Be sure to select the **`End Marker`** radio button before closing the Marker Information dialog.

8 The end marker of a table can be located on or below the rows where table data exists. If your start marker string is on the same row where there may exist table data, then enable the **`Include Marker Line`** option. (The end marker string, **2**, does not appear on the same row as any customer address data. **Do not** enable **`Include Marker Line`**.)

9 If you are certain that your end marker's row contains no other data than the end marker string itself, then enable the **`Select the Line Only Containing This Pattern`** option. (Since your end marker row does contain other data, **do not** enable **`Select the Line Only Containing This Pattern`**.)

CAUTION: Selecting the **`Line Only Containing This Pattern`** will direct the extraction components to search the entire row for verification that no data besides the end marker string exists on that row. To use this option, you must be certain that no other data will ever appear on the end marker row.

10 You can use a row with no data on it as your end marker, by enabling the **`Use Blank Line as Marker`** option. (**Do not** enable **`Use a Blank Line as Marker`**.)

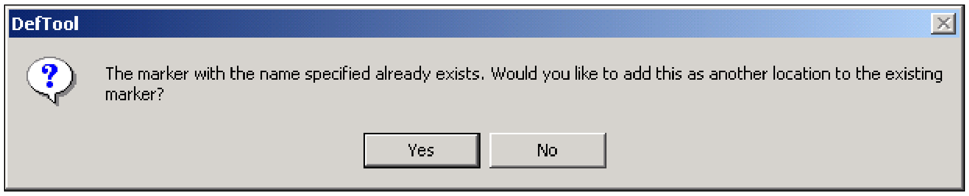
CAUTION: To use a blank line as a marker, you must be certain that a blank row will always indicate the existence of the table, and that there will always be a blank row preceding the table.

11 Select the page style on which the end marker string resides. Your end marker can reside on a different page style than the start marker it is matched with. (Select **`RemitPage`** from the Page Style dropdown list.)

12 Verify that your window coordinates are sufficient to detect your end marker on all statements. If this table will be part of a group, verify that the window coordinates, especially rows, are sufficient to detect ALL occurrences of the end marker on a statement, and across all statements. (Verify that your window coordinates are as follows: Start Row = 10, Start Column = 1, End Row = 14, End Column = 2)

You can create your table rules so that only sections of a table are extracted and presented in a statement, which is driven by the choice of the recipient during statement live retrieval. To do this, you use a dynamic field, which dynamically sets a pattern to a table column, therefore requiring that only data lines where column X = Y will be presented. This feature appears in the Marker Information dialogs to define the same type of rules to extract specific group occurrences. Since markers for groups are created the same way as markers for tables, you see the feature here, however, selective table data extraction and presentment driven by dynamic pattern matching is defined at the table column level.

13 Click **OK** to close the marker Information dialog and save your end marker definition. The marker information dialog closes, and a dialog opens, alerting you to the creation of a marker pair (shown below).



The marker with the name specified already exists. Would you like to add this as another location to the existing marker?

14 Click **Yes** to create a marker pair with your end marker and start marker. In the work area, notice that a yellow area defined by your marker's window coordinate set appears over your end marker string. Also, in the Tree, an End Marker node has been added, creating the marker pair in the tree.

TIP: If you are defining an anchor for the marker used with a table group or a nested table group, make sure the anchor repeats for each occurrence of the group. If the anchor value is not present for an occurrence, that occurrence will not be extracted. The same is true if you are defining an anchor for the marker used with a nested group.

The next task is to define the format of the table. This includes naming the table, selecting the marker pair, defining rules to extract the data as columns and selecting an anchor for the table columns. You must create the marker pair before you can define the table format.

To define the table format:

1 In the Definition Tree, click the Table icon. The Table Format dialog opens (shown below).

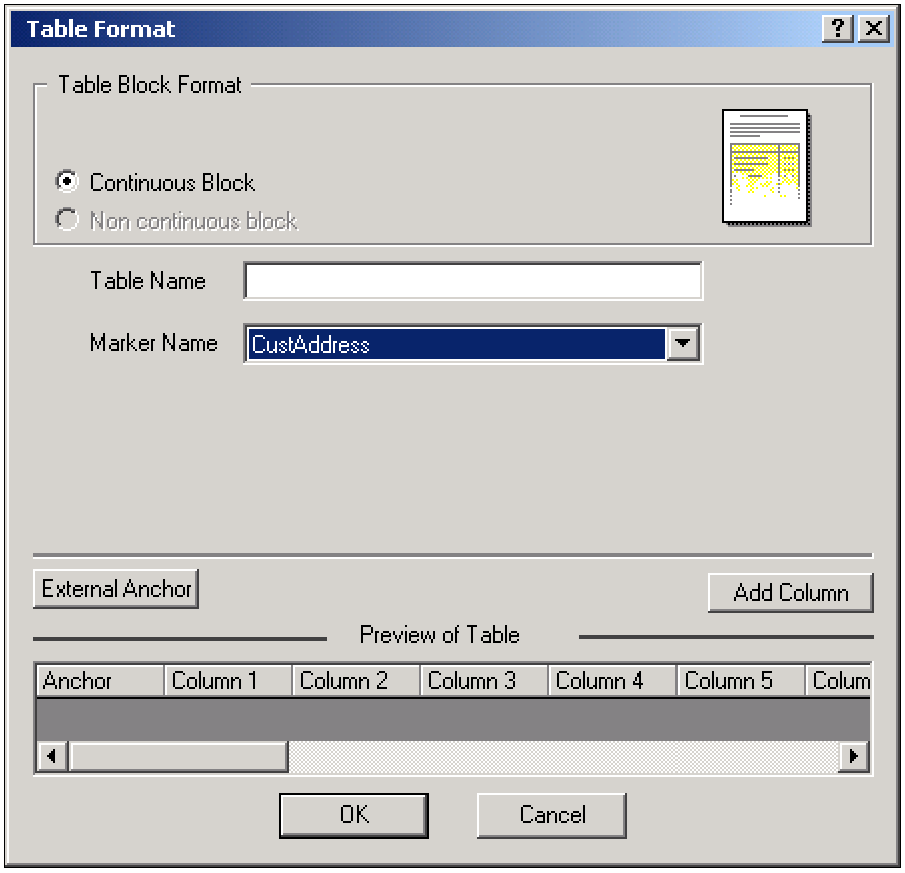


Table Format

Table Block Format

(●) Continuous Block
( ) Non continuous block

Table Name:

Marker Name: CustAddress

2 Enter the table name. Names must not contain spaces or special characters, and must start with an alpha character. The names must not exceed 30 characters in length. (Enter **`CustAddress`**.)

It is a good idea to give your table the same name as the marker pair you're using with it. This helps you organize your tables during presentment definition in the Composer, and also aids in DDF troubleshooting, if necessary.

3. Select the **`Marker End`** from dropdown list, to attach the marker pair to your new table. (Select **`CustAddress`** from the Marker Name dropdown list.)

   CAUTION: **`Do not`** close the Table Format dialog at this point. Your next step is to define the table columns using the Table Format dialog. If you need to close the table Format dialog at any point before you have completed the settings for the table, **`do not`** click **`Cancel`**. This causes you to lose any table format settings you may have made, including the end, markers and any columns created. Rather, click **`OK`**. You can re-enter this dialog from the Tree (right-click the table node and select **`Edit`**) at a later time to complete any work on the table that is left.

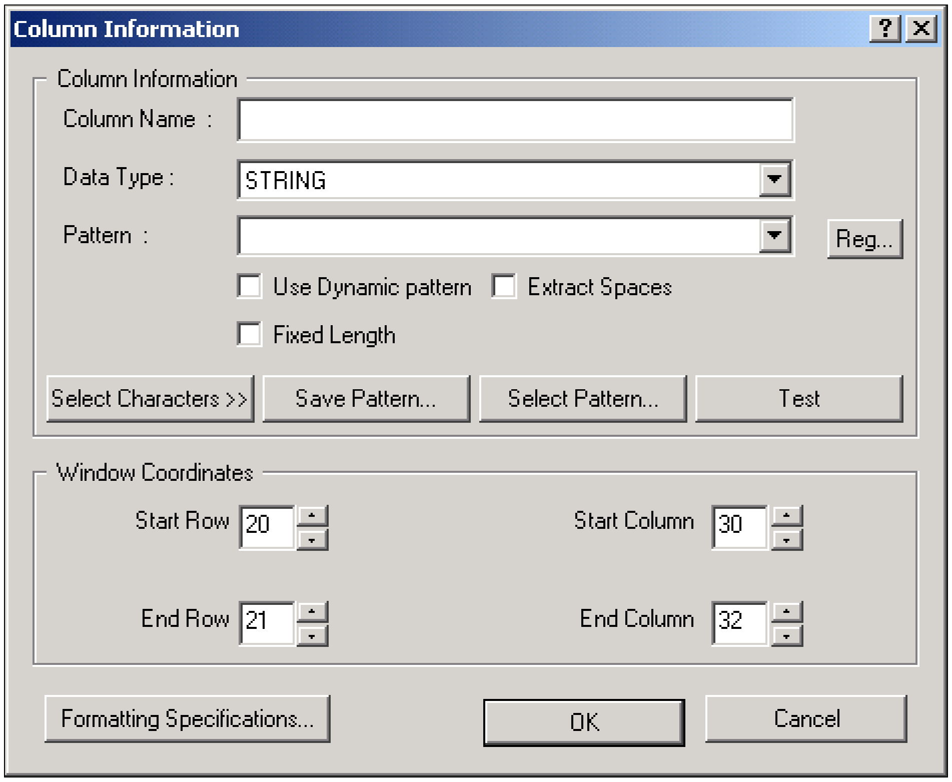
Defining Table Columns

After you have created the table markers, named the table, and selected its markers, your next step is to define the rules to extract the table data as a column or columns. First, you need to identify the columns that will be built into your table, by examining the data samples. (For the CustAddress table, you only need to build one column to extract each line of the data.)

TIP: Generally, you can think of table columns as data elements that iterate **`vertically`**, not horizontally.

To define a table column:

1. Open the Table Format dialog (if not already open) by right-clicking on the Table node in the Tree, and selecting Edit. (You should be working in the Table Format dialog for the CustAddress table.)
2. Near the lower-right corner of the Table Format dialog, click the **`Add Column`** button.
3. Your work area behind the Table Format dialog is live, and will be used as you create the column definition(s). Move the Table Format dialog away from the work area, and navigate in the work area to a view of your table data. Notice that your mouse cursor is a large plus sign **+** when you enable the **`Add Column`** feature.
4. To define your first column, left-click and drag your mouse over the first position of a data string that will be extracted as a column iteration. (On the first statement, select the first **`B`** in **`BILLS BICYCLES`**.) The Column Information dialog opens (shown below).



CAUTION: The definition of a column closely resembles field definition, with one major exception. You **do not** define a span of rows where the column data is to be searched. (Remember, the number of rows that will ultimately be searched for your table columns' data will be dynamically set by the detection of the table start and end markers. Select a start and end row that create an area of only one line.)

5 Enter the column name. The name must start with an alpha character and cannot contain spaces or special characters. Maximum length is 30 characters. (Enter **`CustAddressLine.`**)

6 Enter the pattern that will extract all iterations of the column. (Enter **`[A-Z0-9#%]`** to require that the first position of the address line consists of an alpha-numeric character, a pound sign or a percent sign, since these characters are common to the beginning of an address line.)

As with field definition, your column pattern can be a regular expression that is used to validate and extract the entire string, or it can be used to validate the string by a few characters, while extraction is managed by the Forced Fixed Length.

7 If your column will be extracted by length, click **`Fixed Length`** and set the column length to the desired number of characters to be extracted. (Click **`Fixed Length`**, and in the resulting box to the right, enter 32 as the number of characters to extract.)

8 Verify that the window coordinate set is sufficient to capture the first position of any iteration of the column, including iterations on other statements. Do this by navigating through the statement pages in the work area. (Verify that your window coordinate set is as follows: Start Row = 0, Start Column = 4, End Row = 1, End Column = 5)

9 Make additional Formatting Specifications, if desired, by clicking on the **`Formatting Specifications`** button and selecting post-extraction conversions from the resulting dialog.

10 Click **OK** to close the Column Information dialog. The Column Information dialog closes, and the area defined by the column's window coordinate set is highlighted in aqua. The first column name appears in the Table Format dialog Preview area, under Field 1.

CAUTION: Be sure that you choose a start and end row for your columns that mimic the visual pattern in the data itself. In most cases, the relative column instances occur on the same row (such as the columns of Date, Store and Charge in a Transaction List item), therefore, the definitions for each of these columns must have the same start and end row coordinates.

11 To define each additional column of your table, repeat Steps 2-10.

TIP: Each additional column appears in the Preview area of the Table Format dialog in the order in which you created them. This is the order that they will be presented during live retrieval. If you need to re-order your column, simply right-click the column you want to move, and select Move Left or Move Right until the column is in the desired position.

Defining the Table Anchor

Once you have defined your markers and a table format, including name, marker selection and columns, you are ready to define or select the table anchor. You need to decide at this point whether your table will have:

- **External Anchor** — Based on a data string that iterates with column data if and only if the column data exists, or
- **Internal Anchor** — One of the columns of your table, which is guaranteed to contain data on every row where there is table data.

By default, the first column that is defined becomes the internal anchor of the table. You override this default by either:

- Marking another column as the internal anchor, or
- Defining an external anchor.

(For the *CustAddress* table, there is no external string that iterates with every iteration of column data, therefore we must use a defined column as the Internal anchor. Further, since there is only one column in the *CustAddress* table, this column is the internal anchor by default.)

To define an external anchor:

1. Open the Table Format dialog (if not already open) by right-clicking on the Table node in the Tree, and selecting **Edit**.
2. Near the lower-left corner of the Table Format dialog, click the **External Anchor** button. The Anchor Properties dialog appears.
3. Enter the pattern to validate the anchor string found.
4. Verify that the window coordinates are sufficient (column-wise) to detect the start of the anchor string.
5. **Verify that the row span of the window coordinate set matches the visual pattern of the columns' row spans.** In most cases, choose the same start and end row for the anchor as for the table columns.

6 Click **OK** to close the anchor information dialog. The area defined by the window coordinate set for the anchor is highlighted in pink in the work area.

To select an internal anchor (if the first column is inappropriate):

1 Open the Table Format dialog (if not already open) by right-clicking on the Table node in the Tree, and selecting **`Edit`**.

2 Locate the column which you want to be the internal anchor, in the Preview area of the Table Format dialog.

3 Right-click that column in the Preview area, and select **`Mark as Anchor`**.

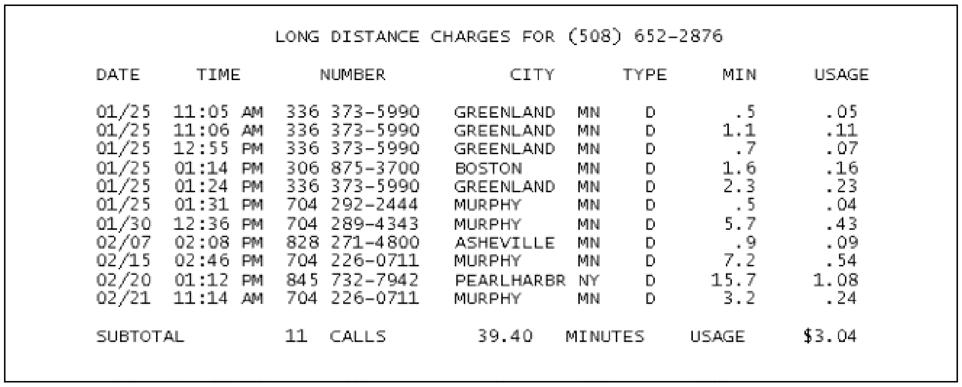
4 Click **OK** to close the Table Format dialog and save the table definition.

9 Defining Groups

When to Use Groups

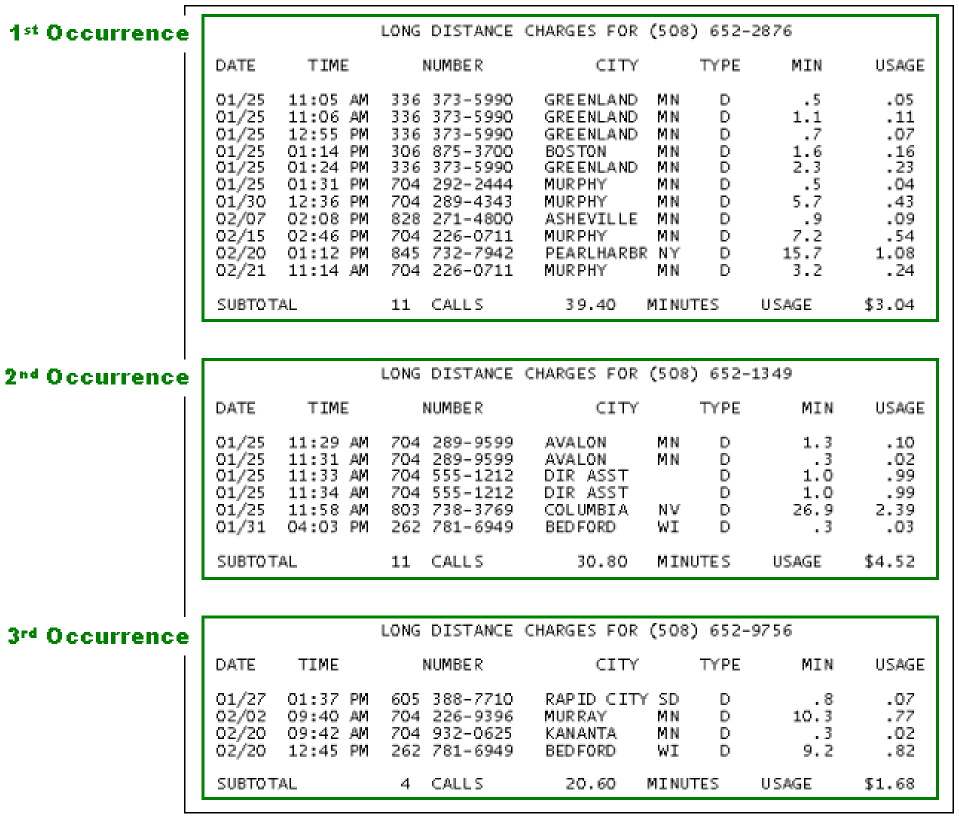
Just as tables can be considered groups of fields (columns) that combine together in a pattern, you can extend this thought to other, more complex visual patterns in your statements. A group is a collection of tables (or data extracted in the form of tables) that iterate together as a pattern.

For example, a telecom company may include a statement section for long distance line charges. This section may contain a title, heading, detail and subtotal (illustrated below):



```
                LONG DISTANCE CHARGES FOR (508) 652-2876

DATE     TIME        NUMBER           CITY        TYPE     MIN      USAGE

01/25  11:05 AM  336 373-5990   GREENLAND  MN    D         .5       .05
01/25  11:06 AM  336 373-5990   GREENLAND  MN    D        1.1       .11
01/25  12:55 PM  336 373-5990   GREENLAND  MN    D         .7       .07
01/25  01:14 PM  306 875-3700   BOSTON     MN    D        1.6       .16
01/25  01:24 PM  336 373-5990   GREENLAND  MN    D        2.3       .23
01/25  01:31 PM  704 292-2444   MURPHY     MN    D         .5       .04
01/30  12:36 PM  704 289-4343   MURPHY     MN    D        5.7       .43
02/07  02:08 PM  828 271-4800   ASHEVILLE  MN    D         .9       .09
02/15  02:46 PM  704 226-0711   MURPHY     MN    D        7.2       .54
02/20  01:12 PM  845 732-7942   PEARLHARBR NY    D       15.7      1.08
02/21  11:14 AM  704 226-0711   MURPHY     MN    D        3.2       .24

SUBTOTAL          11  CALLS         39.40  MINUTES    USAGE      $3.04
```

Assume that customers may have one or more phone lines. In these cases, this type of data, or the long distance charges for each phone line, will be displayed in the statement in the same pattern (an illustration of this repeating pattern follows).


1st Occurrence
2nd Occurrence
3rd Occurrence

```
                    LONG DISTANCE CHARGES FOR (508) 652-2876

DATE     TIME        NUMBER          CITY       TYPE     MIN      USAGE

01/25  11:05 AM  336 373-5990   GREENLAND  MN    D         .5       .05
01/25  11:06 AM  336 373-5990   GREENLAND  MN    D        1.1       .11
01/25  12:55 PM  336 373-5990   GREENLAND  MN    D         .7       .07
01/25  01:14 PM  306 875-3700   BOSTON     MN    D        1.6       .16
01/25  01:24 PM  336 373-5990   GREENLAND  MN    D        2.3       .23
01/25  01:31 PM  704 292-2444   MURPHY     MN    D         .5       .04
01/30  12:36 PM  704 289-4343   MURPHY     MN    D        5.7       .43
02/07  02:08 PM  828 271-4800   ASHEVILLE  MN    D         .9       .09
02/15  02:46 PM  704 226-0711   MURPHY     MN    D        7.2       .54
02/20  01:12 PM  845 732-7942   PEARLHARBR NY    D       15.7      1.08
02/21  11:14 AM  704 226-0711   MURPHY     MN    D        3.2       .24

SUBTOTAL         11  CALLS        39.40   MINUTES     USAGE     $3.04
```

```
                    LONG DISTANCE CHARGES FOR (508) 652-1349

DATE     TIME        NUMBER          CITY       TYPE     MIN      USAGE

01/25  11:29 AM  704 289-9599   AVALON     MN    D        1.3       .10
01/25  11:31 AM  704 289-9599   AVALON     MN    D         .3       .02
01/25  11:33 AM  704 555-1212   DIR ASST         D        1.0       .99
01/25  11:34 AM  704 555-1212   DIR ASST         D        1.0       .99
01/25  11:58 AM  803 738-3769   COLUMBIA   NV    D       26.9      2.39
01/31  04:03 PM  262 781-6949   BEDFORD    WI    D         .3       .03

SUBTOTAL         11  CALLS         30.80   MINUTES     USAGE    $4.52
```

```
                    LONG DISTANCE CHARGES FOR (508) 652-9756

DATE    TIME         NUMBER          CITY       TYPE     MIN      USAGE

01/27  01:37 PM  605 388-7710   RAPID CITY SD    D         .8       .07
02/02  09:40 AM  704 226-9396   MURRAY     MN    D       10.3       .77
02/20  09:42 AM  704 932-0625   KANANTA    MN    D         .3       .02
02/20  12:45 PM  262 781-6949   BEDFORD    WI    D        9.2       .82

SUBTOTAL          4  CALLS        20.60   MINUTES     USAGE     $1.68
```

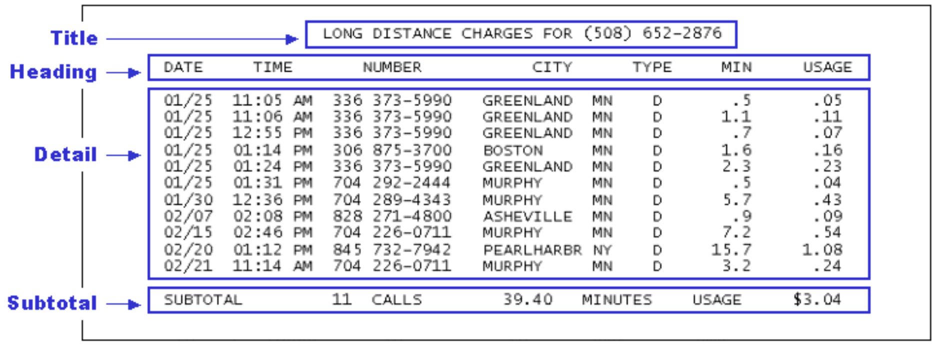
The goal of a group definition is to define the rules to extract a pattern of data once, and allow the extraction components to reuse these rules to detect and extract all other occurrences of the pattern.

You can use the information about the pattern to:

- Build extraction rules to extract each component of the pattern, based on one occurrence of the pattern. **Each "component" of the pattern must be identified properly and extracted as a table.**
- Combine each table into a group definition, so that other occurrences of the tables will be extracted together.

Identifying the Components of a Group Pattern

Building on our illustration above, it is easy to identify the components of this group pattern. To review, this pattern contains a title, heading, some lines of detail and a line of subtotal information. Thus, you should build a table to extract each component. The tables are identified in the illustration below.



```
                LONG DISTANCE CHARGES FOR (508) 652-2876

DATE     TIME        NUMBER            CITY          TYPE    MIN     USAGE

01/25  11:05 AM  336 373-5990  GREENLAND   MN     D       .5      .05
01/25  11:06 AM  336 373-5990  GREENLAND   MN     D      1.1      .11
01/25  12:55 PM  336 373-5990  GREENLAND   MN     D       .7      .07
01/25  01:14 PM  306 875-3700  BOSTON      MN     D      1.6      .16
01/25  01:24 PM  336 373-5990  GREENLAND   MN     D      2.3      .23
01/25  01:31 PM  704 292-2444  MURPHY      MN     D       .5      .04
01/30  12:36 PM  704 289-4343  MURPHY      MN     D      5.7      .43
02/07  02:08 PM  828 271-4800  ASHEVILLE   MN     D       .9      .09
02/15  02:46 PM  704 226-0711  MURPHY      MN     D      7.2      .54
02/20  01:12 PM  845 732-7942  PEARLHARBR  NY     D     15.7     1.08
02/21  11:14 AM  704 226-0711  MURPHY      MN     D      3.2      .24

SUBTOTAL            11   CALLS         39.40   MINUTES     USAGE     $3.04
```

Each table is defined as an ordinary table, each requiring markers, columns and an anchor. When creating tables that will be contained in a group, keep the following important points in mind concerning the table markers:

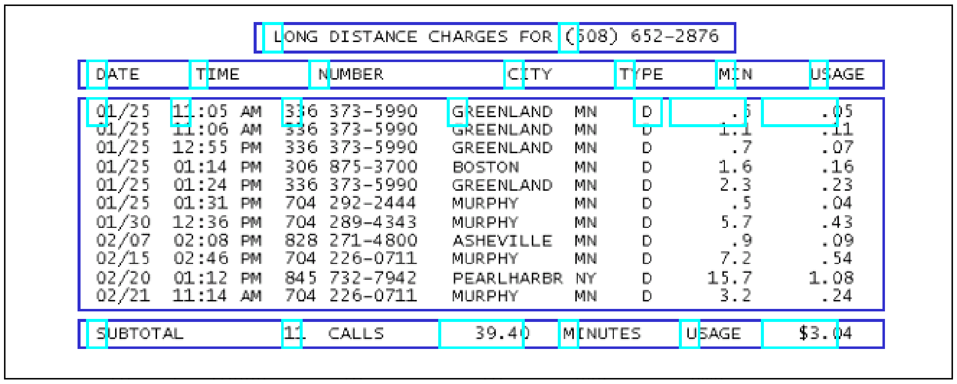
- **Each table marker set must be contained somewhere within the group pattern**
- Each table marker set must iterate with the group occurrences, if there is data for that table in the occurrence.
- Think ahead in terms of the window coordinate set you define. It must be large enough to capture ALL occurrences of the marker pattern, not just on the data example you are building it on.
- Think ahead in terms of the patterns you define for your markers. They must be sufficient to detect ALL iterations of the across group occurrences.
- As with standalone tables, the markers can be defined on or above the table data
- Be sure not to use the same exact string as the start and end marker, as this will create a null searching area for the extraction components.

Table Columns

As with standalone tables, your columns may iterate, however, iteration is not necessary. Good examples of tables whose columns won't iterate in the group occurrence are headings, titles and subtotals. However, a detail table would have columns that iterate in the group. Additional important information about table columns:

- As with standalone tables, you **do not** define your columns' window coordinates to search across all possible rows. During live retrieval, the rows that are searched for column data will be dynamically defined upon the detection of the tables' start and end markers. In short, define your columns on one row only.
- As with standalone tables, the rows that you choose to define your columns on reflect the visual pattern of the column data in the statement. In fact, most table columns (like those of a transaction list or call itemization list) need to be defined on the same row, since the list items iterate together on the same rows.
- Be sure that the patterns for each column are sufficient to capture all iterations of the column, across all occurrences of the group in the statement.

- Building further on the illustrations above, this is a visual depiction of the items identified as table columns (highlighted in aqua) for the long distance line charges, and where you might define each column:



```
                LONG DISTANCE CHARGES FOR (608) 652-2876

DATE        TIME          NUMBER                CITY        TYPE      MIN        USAGE

01/25    11:05  AM   336 373-5990    GREENLAND   MN    D           .6        .05
01/25    11:06  AM   336 373-5990    GREENLAND   MN    D          1.1        .11
01/25    12:55  PM   336 373-5990    GREENLAND   MN    D           .7        .07
01/25    01:14  PM   306 875-3700    BOSTON      MN    D          1.6        .16
01/25    01:24  PM   336 373-5990    GREENLAND   MN    D          2.3        .23
01/25    01:31  PM   704 292-2444    MURPHY      MN    D           .5        .04
01/30    12:36  PM   704 289-4343    MURPHY      MN    D          5.7        .43
02/07    02:08  PM   828 271-4800    ASHEVILLE   MN    D           .9        .09
02/15    02:46  PM   704 226-0711    MURPHY      MN    D          7.2        .54
02/20    01:12  PM   845 732-7942    PEARLHARBR  NY    D         15.7       1.08
02/21    11:14  AM   704 226-0711    MURPHY      MN    D          3.2        .24

SUBTOTAL             11   CALLS         39.40   MINUTES   USAGE        $3.04
```

TIP: By extracting all data from this pattern and avoiding hard-coded titles, headings and labels in your HTML statement templates, you effortlessly control the display of these strings, since they will not be presented unless the relevant data exists in the statement.

To review, groups are combinations of tables that are extracted as a unit for presentment purposes. You will often use groups to extract data in a moderately complex application that contains multiple or combined statement information (such as a consolidated invoice or phone statement) and you want to keep related data elements together when they are presented.

CAUTION: If a table is not part of a group, then the extraction components will carry out ONLY ONE search for the table start and end markers, and thus, only extract ONE occurrence of the table. By creating a group for tables that have multiple occurrences, you order the extraction components to search for as many occurrences of the tables within the group's start and end marker.

Creating a Group

The first step in creating a group is to identify the components of the group. Then, for each component, or table, you follow the steps to create a table.

Select the components of the group. These components can be:

- Tables that you have defined and are part of the group pattern, or
- Inner groups that you have defined, that iterate within the group pattern.

To illustrate each step, we will define a group to extract all occurrences of the *Local Line Detail* sections of a National Wireless statement.

The goal is to extract all of the local line detail information for a phone line in one unit. This will allow us to define presentment for all of the information in the Composer with ease.

You must have identified and built the tables (13 in all) to extract the following *Local Line Detail* information:

`Local Line Phone Number` *Features* section, containing:

- **`Heading`**

- **`Detail Breakdown`**
- **`Subtotal`**

`Subtotal` *Local Usage* section, containing:

- **`Heading`**
- **`Subheading`**
- **`Detail Breakdown`**
- **`Subtotal`**

Itemized Calls section, containing:

- **`Heading`**
- **`Subheading`**
- **`Detail Breakdown`**
- **`Subtotal`**

`Local Line Subtotal`

Each of the above types of information may or may not occur with the group, which we will call **`LocalLineDetailgroup`**. When you recognize a pattern in your statements as described above, you must:

- Build a separate table to extract each "type" of information. This will allow us complete freedom in formatting each of these items uniquely in the Composer.
- Consider any possible item that may appear in the group pattern, and build a table to extract it.

Sometimes, one or more of the items listed above (such as the Itemized Calls section), will not occur with the group pattern on a particular statement. Regardless, you still need to include these items in the group, since they may occur with the group pattern.

CAUTION: You must observer the following rules when creating field, table, group, and marker names:

The names are not sensitive to case, but they must be fashioned from ONLY alphanumeric characters (no special characters are permitted).

The names must start with a letter, and cannot contain any spaces.

The names cannot contain any of the following reserved words alone: Time, Date, INT, Double, Currency. (For example: **`Date`** is not permitted, but **`DueDate`** is allowed.)

Field, table and group names must be unique (marker names are exempt from this requirement).

Keep the names short, e.g. **`TotAmtDue`**. Be descriptive, but use abbreviations instead of long names.

Defining a Group Start Marker

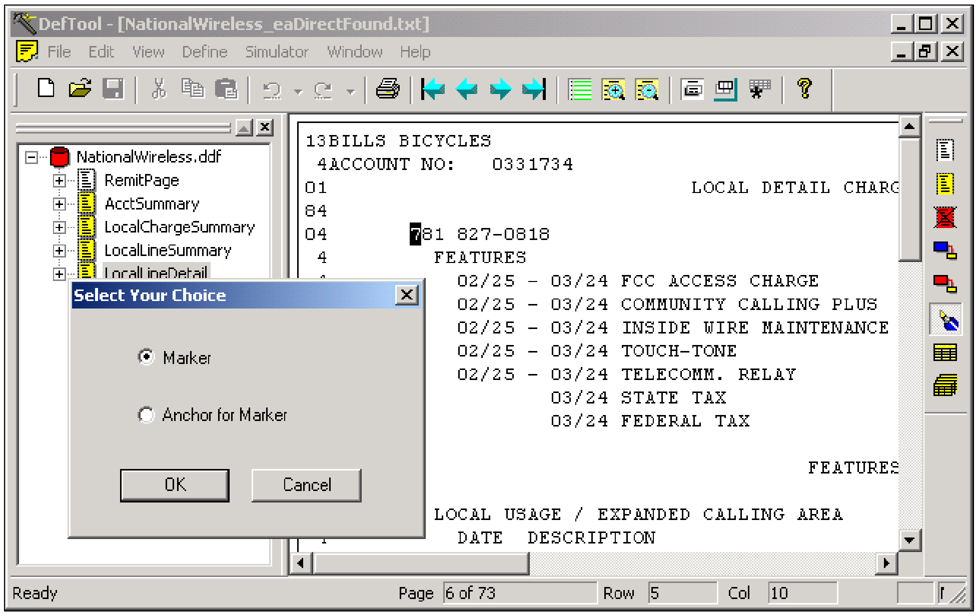
During live retrieval of a statement, a group's start marker is used to inform the extraction components of the row on which to start searching for group data. You start the process of defining the start marker by first identifying a data string that appears on or above the row where group data begins. Start markers are mandatory, however, you may define more than one start marker for a group.

To define a group start marker:

1. Navigate to a LocalLineDetail page of any National Wireless statement in the DefTool work area.
2. From the Definition Tool Bar, select the Marker icon.
3. Identify and locate the data string that indicates the start row of the group's searching area. This is where the extraction components will begin using the group's table and inner group definitions (markers, columns, etc.) to extract your new group's data. (Since there is no non-data string that occurs on or above every LocalLineDetail group occurrence, use the phone number that always occurs at the start of the group. Locate any phone number between columns 9-10.)
4. Select the Marker icon from the Definition Tool Bar.

   TIP: You can easily find out what coordinates your mouse is on. If you move your mouse into the work area, and then look at the Status Bar in the bottom pane of the screen, you can see the row and column coordinates of the cursor's location.

5. Move your mouse into the work area and left-click drag and drop your mouse over the first position of the string identified in Step 2. (Select the first numeric character in the phone number as shown below.) The Select Your Choice dialog opens.



```
13BILLS BICYCLES
 4ACCOUNT NO:   0331734
01                                  LOCAL DETAIL CHARG
84
04        781 827-0818
 4           FEATURES
               02/25 - 03/24 FCC ACCESS CHARGE
               02/25 - 03/24 COMMUNITY CALLING PLUS
               02/25 - 03/24 INSIDE WIRE MAINTENANCE
               02/25 - 03/24 TOUCH-TONE
               02/25 - 03/24 TELECOMM. RELAY
                       03/24 STATE TAX
                       03/24 FEDERAL TAX

                                                FEATURES

             LOCAL USAGE / EXPANDED CALLING AREA
               DATE  DESCRIPTION
```

It is sufficient at this point to just select the first character (shown above). You must edit the window coordinates later in the Marker Information dialog to be large enough to capture other Phone Numbers.

6. From here, decide if you are creating a marker or an anchor for the marker (which you would create immediately after the anchor). (Accept the default of marker, and click **OK**.) The Marker Information dialog opens.

7 Enter the marker end. Keep names short but descriptive, as in TotAmtDue. Field, table, group, and marker names must start with a letter, use alphanumeric characters only (no special characters), and cannot contain spaces. Do not use the following reserved words in a name: Time, Date, INT, Double, Currency ("Date" is not permitted but "DueDate: is). Field, table and group names (but not marker names) must be unique. Maximum length is 30 characters. Names are not case-sensitive. (Enter **`LocalLineDetailgroup.`**)

8 Enter the pattern to be searched for as the start marker. Be sure that your pattern is sufficient to detect this start marker on all statements. Enter a regular expression to extract the phone number as the pattern. You should have already created this regular expression for use in other tables and columns. The pattern to extract the phone number is:

```
[0-9][0-9][0-9] [0-9][0-9][0-9]\-[0-9][0-9][0-9][0-9]
or \n\n\n \n\n\n\-\n\n\n\n
```

9 When creating a marker, the Start Marker radio button is selected by default. Accept this default for the start marker definition. (Accept the default radio selection of Start Marker.)

10 The start marker of a group can be located on or above the rows where group data exists (including markers of tables and inner groups). If your group start marker string is on the same row where there may exist table data, and/or used as a marker for an inner table or group, then enable the **`Include Marker Line`** option. (The start marker string, or the phone number, is part of the group data, and is also used as the start marker for tables within the group. Click to enable **`Include Marker Line`**.)

11 The start marker string can be "re-used" as the end marker, meaning that you can define the end marker on the same exact string in the same location, with the intent that it be detected on a subsequent page. If you want to use the same string for the start and end marker and NOT create a null searching area for the group data, then enable **`Look for the Start Marker on the Same Row`** in the Start Marker Information dialog. (Since your start marker and end marker will be different for the LocalLineDetailGroup, **do not** enable "**`Look for the Start Marker on the Same Row`**.")

12 If you are certain your start marker's row contains no other data than the start marker string itself, enable the **`Select the Line Only Containing This Pattern`** option. (Since your start marker row does contain other data, **do not** enable **"`Select the Line Only Containing This Pattern`**.")

CAUTION: **"`Select the Line Only Containing This Pattern`"** directs the extraction components to search the entire row for verification that no data besides the start marker string exists on that row. To use this option, you must be certain that no other data will ever appear on the start marker row.

13 You can use a row with no data on it as your start marker, by enabling the **`Use Blank Line as Marker`** option. (For the National Wireless application, **do not** enable **"`Use a Blank Line as Marker`**.")

CAUTION: To use a blank line as a marker, you must be certain that a blank row will always indicate the existence of the table, and that there will always be a blank row preceding the table.

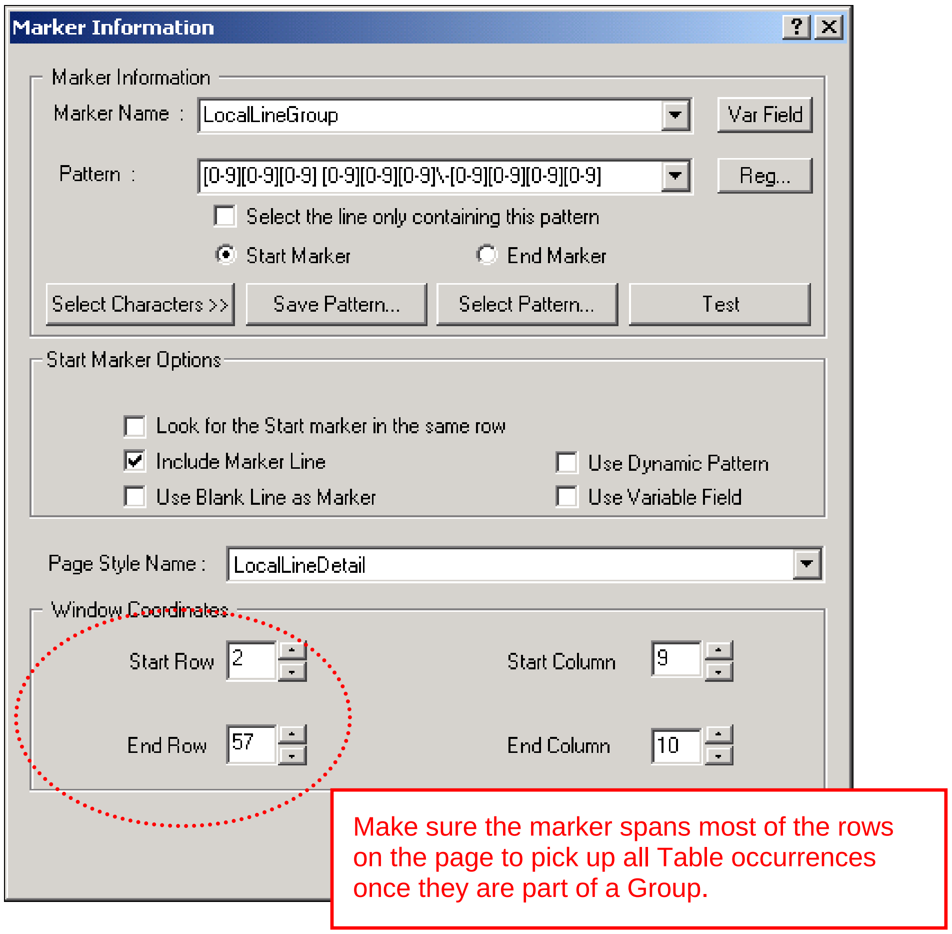
14 Select the **`page style`** on which the start marker string resides. (Select **`LocalLineDetail`** from the Page Style dropdown list.)

15 Verify that your window coordinates are sufficient to detect your group start marker on all statements. Since the phone number can appear near the top or bottom of the page, your row

coordinates will span much of the page. Verify that your window coordinates are: Start Row = 2, Start Column = 9, End Row = 57, End Column = 10.

TIP: You can create your group rules so that only certain occurrences of a group are extracted and presented in a statement, which is driven by the choice of the recipient during statement live retrieval. To do this, you insert a dynamic field in the pattern, which dynamically sets a pattern to a group marker, therefore requiring that only data where group marker = X will be presented. Dynamic pattern matching for selective group extraction is discussed in “Setting Dynamic Pattern Markers.”

The completed Marker Information dialog appears as follows:



Marker Information

Marker Name : LocalLineGroup

Pattern : [0-9][0-9][0-9] [0-9][0-9][0-9]\-[0-9][0-9][0-9][0-9]

☐ Select the line only containing this pattern

[✓ Start Marker] End Marker

☐ Look for the Start marker in the same row

☑ Include Marker Line

☐ Use Blank Line as Marker

☐ Use Dynamic Pattern

☐ Use Variable Field

Page Style Name : LocalLineDetail

Start Row 2

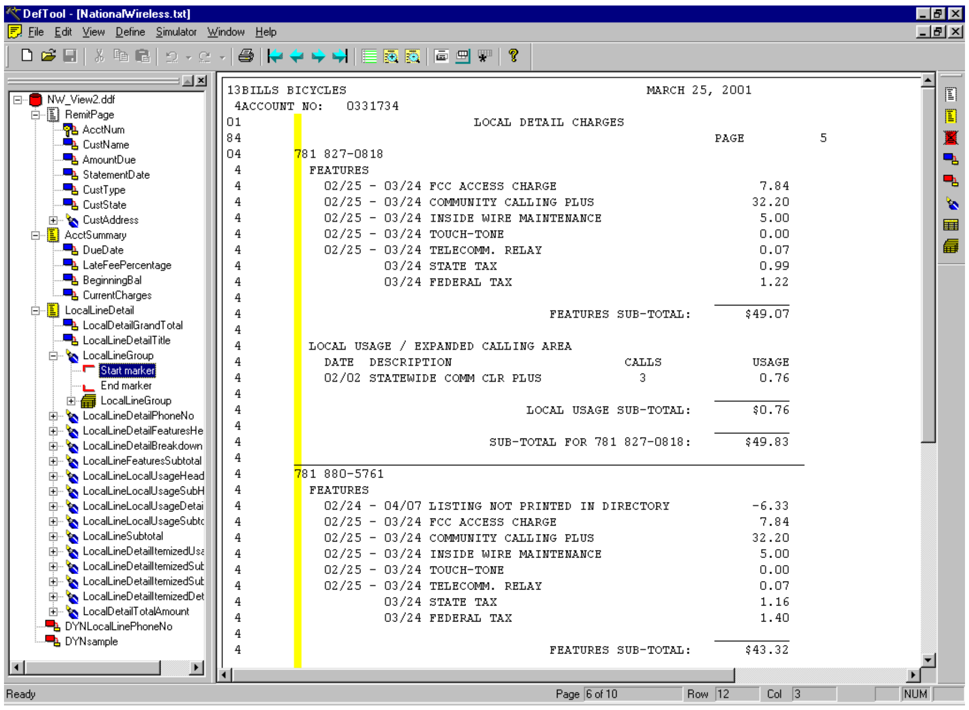
End Row 57

Start Column 9

End Column 10

Make sure the marker spans most of the rows on the page to pick up all Table occurrences once they are part of a Group.

16 Click **OK** to close the Marker Information dialog and save your group start marker definition. In the work area, notice that a yellow area defined by your marker’s window coordinate set appears over your start marker string. Also, in the Tree, a Start Marker node has been added to the tree. This is shown below.



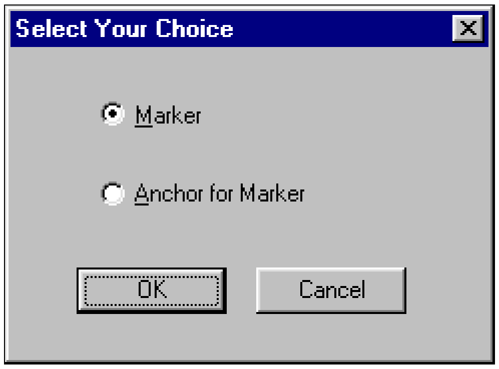
```
13BILLS BICYCLES                                          MARCH 25, 2001
 4ACCOUNT NO:   0331734
01                                 LOCAL DETAIL CHARGES
84                                                                 PAGE          5
04         781 827-0818
 4           FEATURES
 4             02/25 - 03/24 FCC ACCESS CHARGE                              7.84
 4             02/25 - 03/24 COMMUNITY CALLING PLUS                        32.20
 4             02/25 - 03/24 INSIDE WIRE MAINTENANCE                        5.00
 4             02/25 - 03/24 TOUCH-TONE                                     0.00
 4             02/25 - 03/24 TELECOMM. RELAY                                0.07
 4                     03/24 STATE TAX                                      0.99
 4                     03/24 FEDERAL TAX                                    1.22
 4                                                                  _________
 4                                     FEATURES SUB-TOTAL:             $49.07
 4
 4           LOCAL USAGE / EXPANDED CALLING AREA
 4             DATE  DESCRIPTION                       CALLS              USAGE
 4             02/02 STATEWIDE COMM CLR PLUS             3                 0.76
 4                                                                  _________
 4                                  LOCAL USAGE SUB-TOTAL:              $0.76
 4                                                                  _________
 4                              SUB-TOTAL FOR 781 827-0818:            $49.83
 4         ____________________________________________________________________
 4         781 880-5761
 4           FEATURES
 4             02/24 - 04/07 LISTING NOT PRINTED IN DIRECTORY              -6.33
 4             02/25 - 03/24 FCC ACCESS CHARGE                              7.84
 4             02/25 - 03/24 COMMUNITY CALLING PLUS                        32.20
 4             02/25 - 03/24 INSIDE WIRE MAINTENANCE                        5.00
 4             02/25 - 03/24 TOUCH-TONE                                     0.00
 4             02/25 - 03/24 TELECOMM. RELAY                                0.07
 4                     03/24 STATE TAX                                      1.16
 4                     03/24 FEDERAL TAX                                    1.40
 4                                                                  _________
 4                                     FEATURES SUB-TOTAL:             $43.32
```

Defining a Group End Marker

During live retrieval of a statement, a group's end marker is used to inform the extraction components of the row on which to stop searching for group data. You start the process of defining the end marker similarly to a start marker, by first identifying a data string that appears on or below the row where table data is expected to end.

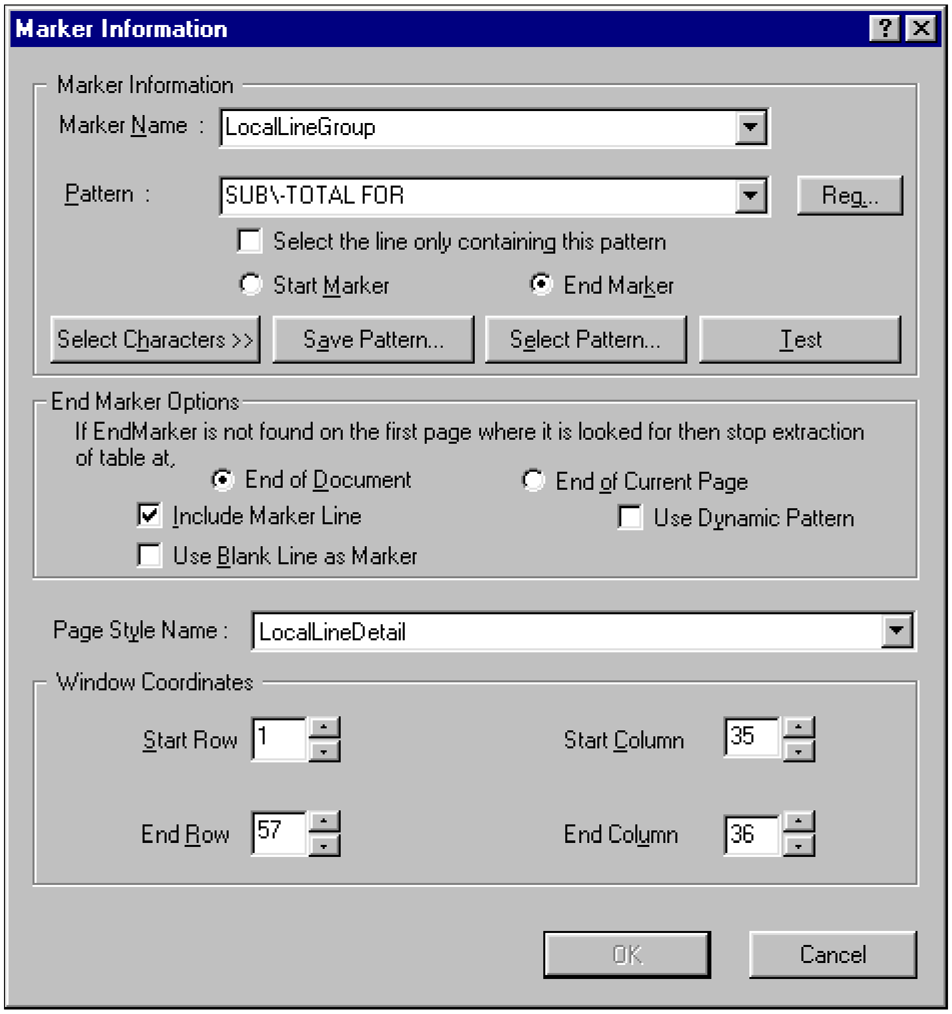
To define a group end marker:

1. Identify and locate the data string that indicates the end row of the group's searching area. This is where the extraction components will stop looking for column data. (The string *SUB-TOTAL FOR*, starting between columns 35-36, is the best indicator of the end of the LocalLineDetail for a Phone Number (or the group occurrence)).
2. Select the Marker icon from the Definition Tool Bar .
3. Move your mouse into the work area and left-click drag and drop your mouse over the first position of the string identified in Step 1. (Select the first character **`S`** of **`SUB-TOTAL FOR`** between columns 35-36.) The Select Your Choice dialog opens.


Select Your Choice

(●) Marker
( ) Anchor for Marker

4 Decide if you are creating a marker or an anchor for the marker (which you would create immediately after the anchor). (Accept the default of **Marker**, and click **OK**.) The Marker Information dialog opens (shown below).


Marker Information

Marker Information
Marker Name : LocalLineGroup
Pattern : SUB\-TOTAL FOR
[ ] Select the line only containing this pattern
( ) Start Marker (●) End Marker

End Marker Options
If EndMarker is not found on the first page where it is looked for then stop extraction of table at,
(●) End of Document ( ) End of Current Page
[✓] Include Marker Line [ ] Use Dynamic Pattern
[ ] Use Blank Line as Marker

Page Style Name : LocalLineDetail

Window Coordinates
Start Row 1 Start Column 35
End Row 57 End Column 36

5 Enter the marker end. If you have already created and named the start marker that will be used with this new end marker, then you already have the end marker name. Select this name from the Marker Name dropdown list or enter it manually. Names must not contain spaces or special

characters, must start with an alpha character, and cannot exceed 30 characters in length. (Select **LocalLineDetailgroup.)**

6 Enter the pattern to be searched for as the end marker. Be sure that your pattern is sufficient to detect this start marker on all statements. (Enter **SUB\-TOTAL FOR** as the pattern.)

7 When creating any marker, the **Start Marker** radio button is selected by default. Be sure to select the **End Marker** radio button before closing the Marker Information dialog. (Click to select the End Marker radio button.)

8 The end marker of a group can be located on or below the rows where group data exists. If your group end marker string is on the same row where there may exist table data, then enable the **Include Marker Line** option. (The end marker string, *SUB-TOTAL FOR*, does appear on the same row as group data, and is used as an end marker for some inner tables. Click to enable the **Include Marker Line** option.)

9 If you are certain that your end marker's row contains no other data than the end marker string itself, then enable the **Select the Line Only Containing This Pattern** option. (Since your end marker row does contain other data, **do not** enable **Select the Line Only Containing This Pattern**.)

CAUTION: **Select the Line Only Containing This Pattern** will direct the extraction components to search the entire row for verification that no data besides the end marker string exists on that row. To use this option, you must be certain that no other data will ever appear on the end marker row.

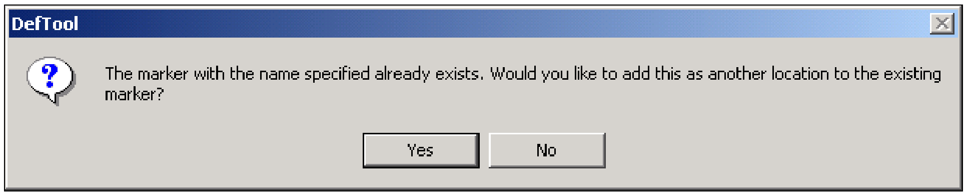
10 You can use a row with no data on it as your end marker, by enabling the **Use Blank Line as Marker** option. (**Do not** enable Use a Blank Line as Marker.)

CAUTION: To **Use a Blank Line as a Marker**, you must be certain that a blank row will always indicate the existence of the group, and that there will always be a blank row preceding the group.

11 Select the page style on which the end marker string resides. Select **LocalLineDetail** from the Page Style dropdown list. Verify that your window coordinates are: Start Row = 1, Start Column = 35, End Row = 57, End Column = 36.

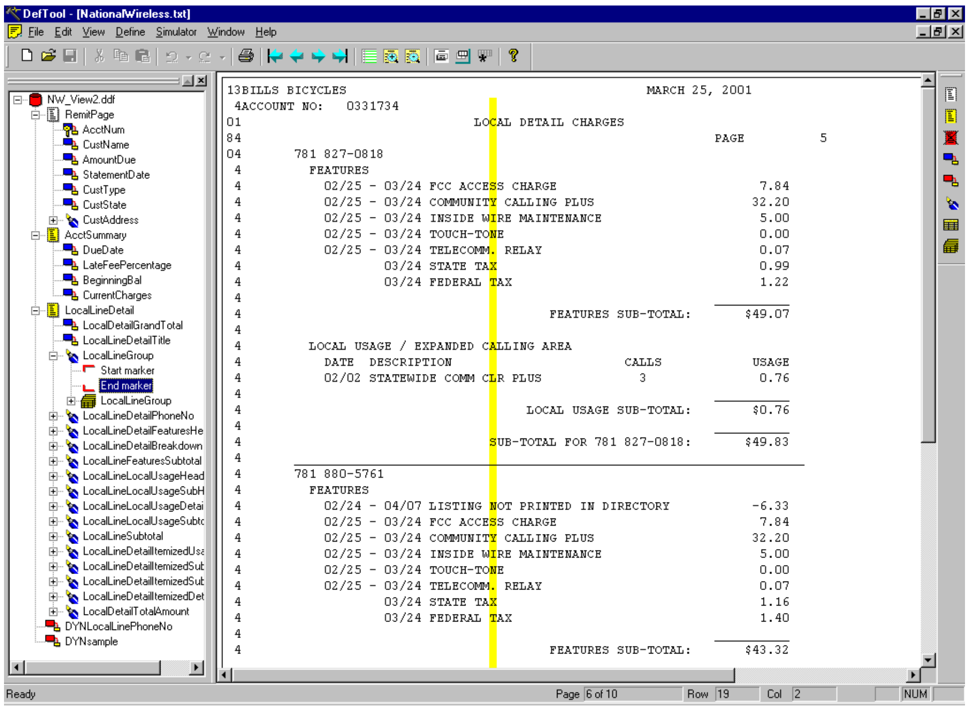
TIP: Your end marker can reside on a different page style than the start marker it is matched with.

12 Click **OK** to close the marker Information dialog and save your end marker definition. The marker Information dialog closes, and a dialog opens, alerting you to the creation of a marker pair (shown below).



The marker with the name specified already exists. Would you like to add this as another location to the existing marker?

13 Click **Yes** to create a marker pair with your group end marker and group start marker. In the work area, notice that a yellow area defined by your group end marker's window coordinate set appears over your group end marker string, and also spans most rows on the page. Also, in the Tree, an End Marker node has been added, creating the Marker pair in the tree (shown in the figure that follows).


DefTool - [NationalWireless.txt]
File Edit View Define Simulator Window Help
Ready
Page 6 of 10
Row 19
Col 2
NUM

```
NW_View2.ddf
  RemitPage
    AcctNum
    CustName
    AmountDue
    StatementDate
    CustType
    CustState
    CustAddress
  AcctSummary
    DueDate
    LateFeePercentage
    BeginningBal
    CurrentCharges
  LocalLineDetail
    LocalDetailGrandTotal
    LocalLineDetailTitle
    LocalLineGroup
      Start marker
      End marker
      LocalLineGroup
    LocalLineDetailPhoneNo
    LocalLineDetailFeaturesHe
    LocalLineDetailBreakdown
    LocalLineFeaturesSubtotal
    LocalLineLocalUsageHead
    LocalLineLocalUsageSubH
    LocalLineLocalUsageDetai
    LocalLineLocalUsageSubtc
    LocalLineSubtotal
    LocalLineDetailItemizedUsa
    LocalLineDetailItemizedSub
    LocalLineDetailItemizedSub
    LocalLineDetailItemizedDet
    LocalDetailTotalAmount
  DYNLocalLinePhoneNo
  DYNsample
```

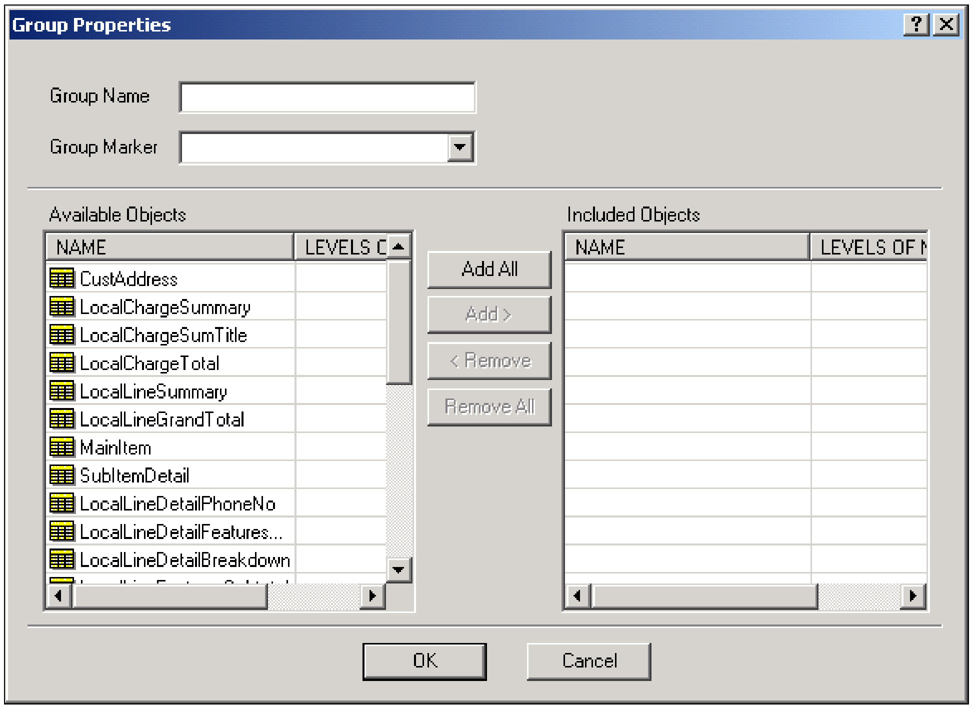
```
13BILLS BICYCLES                                         MARCH 25, 2001
 4ACCOUNT NO:   0331734
01                                LOCAL DETAIL CHARGES
84                                                             PAGE         5
04      781 827-0818
 4        FEATURES
 4          02/25 - 03/24 FCC ACCESS CHARGE                                7.84
 4          02/25 - 03/24 COMMUNITY CALLING PLUS                          32.20
 4          02/25 - 03/24 INSIDE WIRE MAINTENANCE                          5.00
 4          02/25 - 03/24 TOUCH-TONE                                       0.00
 4          02/25 - 03/24 TELECOMM. RELAY                                  0.07
 4                  03/24 STATE TAX                                        0.99
 4                  03/24 FEDERAL TAX                                      1.22
 4                                                               ________
 4                                        FEATURES SUB-TOTAL:         $49.07
 4
 4        LOCAL USAGE / EXPANDED CALLING AREA
 4          DATE  DESCRIPTION                          CALLS           USAGE
 4          02/02 STATEWIDE COMM CLR PLUS                3             0.76
 4                                                               ________
 4                                     LOCAL USAGE SUB-TOTAL:          $0.76
 4                                                               ________
 4                                 SUB-TOTAL FOR 781 827-0818:        $49.83
 4      ____________________________________________________________________
 4      781 880-5761
 4        FEATURES
 4          02/24 - 04/07 LISTING NOT PRINTED IN DIRECTORY                -6.33
 4          02/25 - 03/24 FCC ACCESS CHARGE                                7.84
 4          02/25 - 03/24 COMMUNITY CALLING PLUS                          32.20
 4          02/25 - 03/24 INSIDE WIRE MAINTENANCE                          5.00
 4          02/25 - 03/24 TOUCH-TONE                                       0.00
 4          02/25 - 03/24 TELECOMM. RELAY                                  0.07
 4                  03/24 STATE TAX                                        1.16
 4                  03/24 FEDERAL TAX                                      1.40
 4                                                               ________
 4                                        FEATURES SUB-TOTAL:         $43.32
```

Defining Group Properties

The next task is to define the properties of the group. This includes naming the group, selecting the marker pair, and selecting the data items that will be searched for and extracted as part of the group. To review, the data items that you can include as part of a group are tables and inner groups. You cannot include fields as part of a group. You must create the marker pair before you can define the group properties.

To define group properties:

1 From the Definition Tool Bar, click the Group icon. The Group Properties dialog opens.



2 In the Group Properties dialog, enter the group name in the corresponding field. (Enter **`LocalLineDetailGroup`**.)

TIP: Just as for tables, it is a good idea to give your group the same name as the marker pair that will be used with it. This will help you organize your groups during presentment definition in the Composer, and also aid in DDF troubleshooting, if necessary.

3 Select the group marker from the corresponding dropdown list, to attach the marker pair to your new group. (Select **`LocalLineDetailGroup`** from the group marker dropdown list.)

4 Your next step is to define the group's objects. All of the tables and groups that you have created in your DDF will be displayed in the Available Objects section of the Group Properties dialog. Select the object that you want to include in the group, from the Available Objects section, and click **`Add >`**. The object will appear under the Included Objects section. (Select the following tables that you have built to extract items from the Local Line Detail statement sections:)

`LocalLineDetailPhoneNo`

`LocalLineDetailFeaturesHead`

`LocalLineDetailFeaturesBreakdown`

`LocalLineDetailFeaturesSubtotal`

`LocalLineDetailLocalUsgHead`

`LocalLineDetailLocalUsgSubhead`

`LocalLineDetailLocalUsgBreakdn`

`LocalLineDetailLocalUsgSubtotal`

`LocalLineDetailItemizedHead`

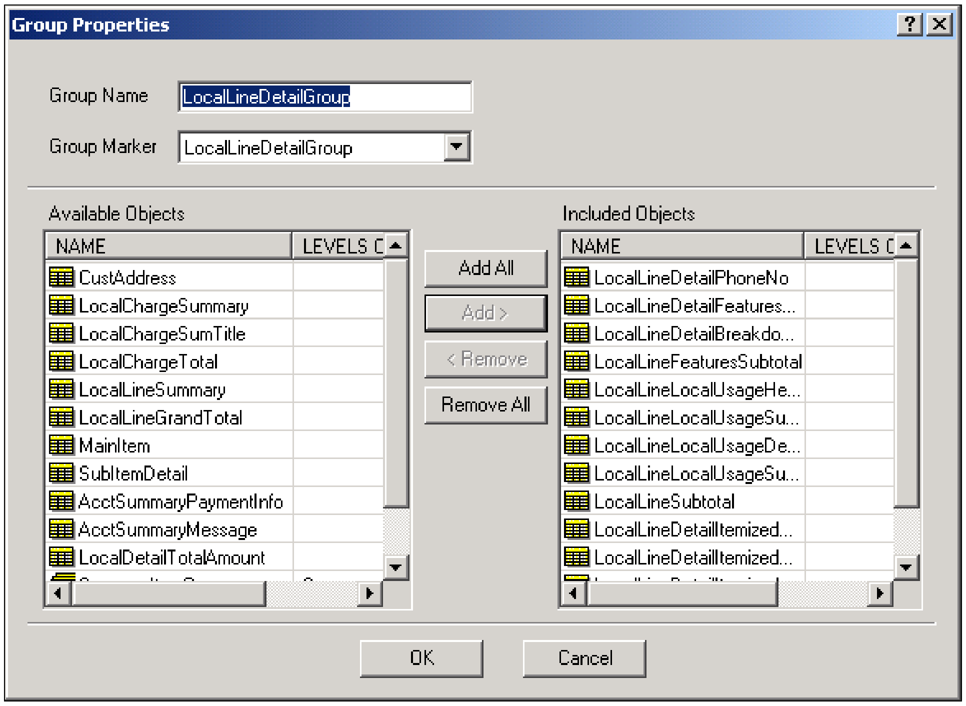
`LocalLineDetailItemizedSubhead`

`LocalLineDetailItemizedBreakdn`

`LocalLineDetailItemizedSubtotal`

`LocalLineDetailSubtotal`

TIP: You can also double-click an object in the **`Group Properties Available Objects`** list to add it to the group. To add ALL available tables and groups to your new group, click **`Add All`**.



Group Name: LocalLineDetailGroup
Group Marker: LocalLineDetailGroup

Available Objects: CustAddress, LocalChargeSummary, LocalChargeSumTitle, LocalChargeTotal, LocalLineSummary, LocalLineGrandTotal, MainItem, SubItemDetail, AcctSummaryPaymentInfo, AcctSummaryMessage, LocalDetailTotalAmount

Included Objects: LocalLineDetailPhoneNo, LocalLineDetailFeatures..., LocalLineDetailBreakdo..., LocalLineFeaturesSubtotal, LocalLineLocalUsageHe..., LocalLineLocalUsageSu..., LocalLineLocalUsageDe..., LocalLineLocalUsageSu..., LocalLineSubtotal, LocalLineDetailItemized..., LocalLineDetailItemized...

5 Click **OK** to close the Group Properties dialog and save the new group settings.

About available objects and nested groups

All tables and groups will appear in the Available Objects list. This can be confusing if you are creating nested groups.

CAUTION: If creating nested groups, **do not** select tables that are part of a group.

For example, if you want to build a nested group to include three tables and one inner group (containing five tables), the three tables, the inner group, **AND** the five tables of the inner group appear in Available Objects. The correct selection will be the three tables and the innergroup. **Do not** select the five tables of the inner group, because they are already grouped.

10 Presenting Sub-Account Data

When to Use Sub-Account Indexing

The eStatement Manager sub-account indexing feature lets you create a view specifically for presenting sub-account data. Use sub-account indexing if you need to:

- Quickly access sub-account data in a data input file for presentment in a view.
- Include sub-account information in a history list, or "hit list."

For example, you could define the cell phone detail sections of a large B2B telecommunications statement as sub-accounts, speeding Web access to line detail on request.

You must create a separate view (DDF/ALF pair) for presenting each view of sub-account information to users. You can also define more than one sub-account for an application. The larger document that a sub-document belongs to is called the root, or parent, document. One parent document can consist of many instances of one or more sub-documents (sometimes referred to as the "children" of the parent document).

To enable the sub-account indexing feature, you must define the sub-account data area of an input file as a group, then define a group field on a piece of data that uniquely identifies the sub-account (or sub-document), called a subkey. Just as a primary key identifies a document, the subkey identifies each sub-document group. You can define additional group fields for presentment as well as for indexing.

You must also define the sub-account groups and any group fields you want to index in the *application's indexing DDF* (you can have only one indexing DDF for an entire application). Index data enables you to include sub-accounts in a history list and to click a link to quickly access sub-account data in a view (without having to scan the entire statement to find the group information).

The following section describes the general process you must follow to set up sub-document indexing in your application.

Sample Sub-Document Indexing Application

eStatement Manager includes the following sample application files to demonstrate sub-document indexing.

A sample DDF for indexing sample sub-document data:

- *<EDX_HOME>\samples\NW_Subdocument\IndexerJob\NW_SubdocumentIndexer.ddf*

(Note that a separate DDF for indexing sub-documents is included here only for demonstration purposes; in your application, create one DDF for indexing all documents and sub-documents.)

Sample sub-document view files:

- *<EDX_HOME>\samples\NW_Subdocument\NW_Subdocument.ddf*
- *<EDX_HOME>*\samples\NW_Subdocument\NW_Subdocument.alf

Use the *NationalWireless.txt* data file with this application.

Setting up Sub-Account Indexing

This section describes the general process required to implement sub-document indexing. You must tailor these procedures to the particular data in your input file and the views you need to present.

Define the sub-document data as a **group**. (This includes creating group markers, creating the group, and assigning the group markers to a group.)

You must identify a piece of data that uniquely identifies each sub-account in the data input file, such as sub-account number, and define this as the **subkey** group field for the sub-account. The subkey must be located within the sub-account group.

eStatement Manager uses the subkey to locate the requested sub-document during dynamic retrieval. When the Indexer job indexes the subkey, it stores the exact offsets to each sub-document. By storing offsets for each sub-document, eStatement Manager has the information it needs to directly access any sub-document in the input file, eliminating the need to search the entire data file to identify the sub-document boundary.

Consider sub-documents in the overall plan, design, and implementation of your eStatement Manager application.

To implement sub-account indexing:

- Define the sub-account groups and any group fields you want indexed, such as the name or other information associated with the sub-account number, **in your application's indexing DDF**. eStatement Manager automatically indexes all fields defined in the DDF that you publish with the Indexer job.
- Create custom DDF and ALF view files for presenting your sub-document views. You may need one or more DDFs for sub-document presentment, but each different view requires a custom ALF.

See "Defining Sub-Account Groups, Subkey, and Group Fields" for details about defining sub-documents in your indexing and presentment DDFs.

Defining Sub-Account Groups, Subkey, and Group Fields in a DDF

Follow the steps below to define the sub-document group, subkey, and group fields in the application's indexing DDF and in the custom DDFs you must create for sub-document presentment. You can define more than one group field for a group, but you must define only one as the subkey.

Be sure to add any group fields to the indexing DDF that you want indexed.

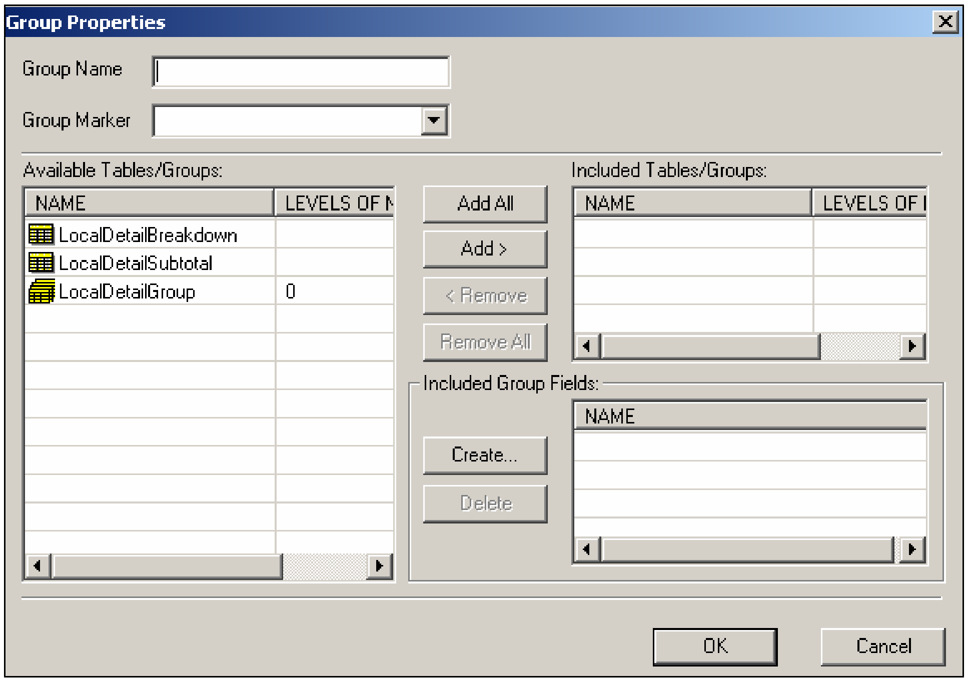
Define group fields within the sub-account parent group. You can also define group fields in nested groups.

To define a sub-account group, subkey, and any group fields in the application's indexing DDF or in a sub-account presentment DDF:

1 In DefTool, open the DDF (or, to create a new presentment DDF, select **`File>New`** and complete the Data Configuration Wizard. Complete the wizard and follow the procedures described in this guide for creating a document style, primary key, and any additional page styles).

2 Click the New Marker icon and define the start and end markers for the sub-account group. You define the start and end markers for the group indicating the group offsets – the exact location in the data input file where eStatement Manager should begin searching for the group data. Once you define a group, it appears in the Definition Tree on the left of the DefTool window for the DDF.

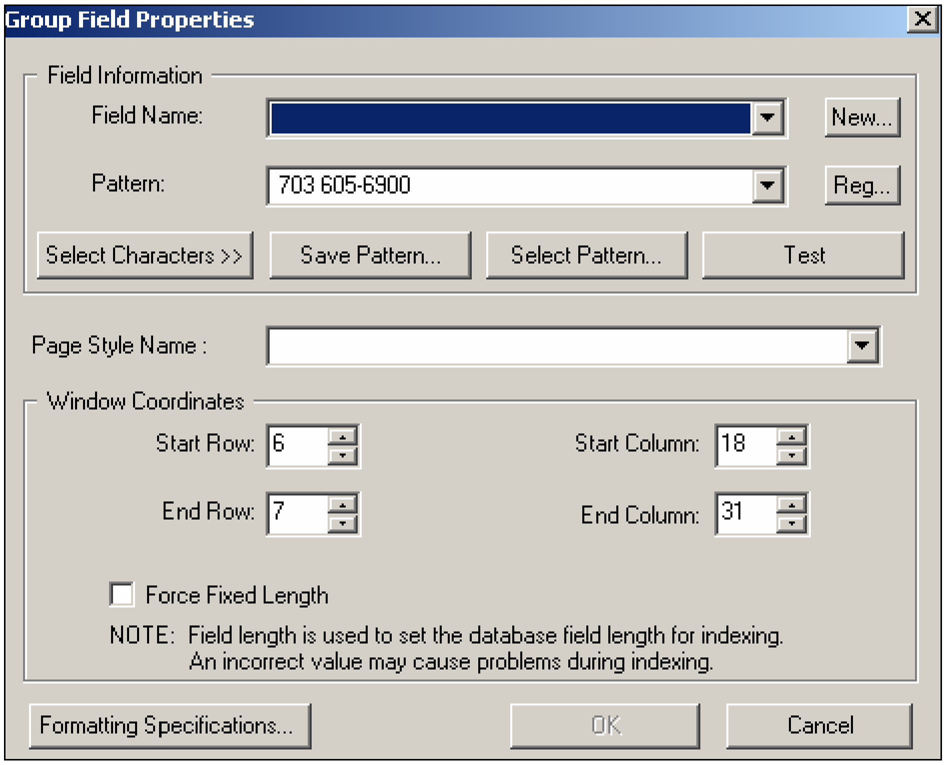
3 Click the Group icon to display the Group Properties dialog. Define the properties of the sub-account group. Specify the group name, associated markers, and select the tables in the group.



NAME	LEVELS OF N
LocalDetailBreakdown	
LocalDetailSubtotal	
LocalDetailGroup	0

4 Create a subkey for the sub-documents; identify a piece of data in the data input file that uniquely identifies each sub-account. The subkey and any anchor you specify must be located between the group's start and end markers.

5 On the Group Properties dialog, click **Create**. Use the left mouse button to drag the crosshair selector box around the data you want to use for the subkey group field. At the Select Your Choice dialog, select **Field** and click **OK**. DefTool displays the Group Field Properties dialog:



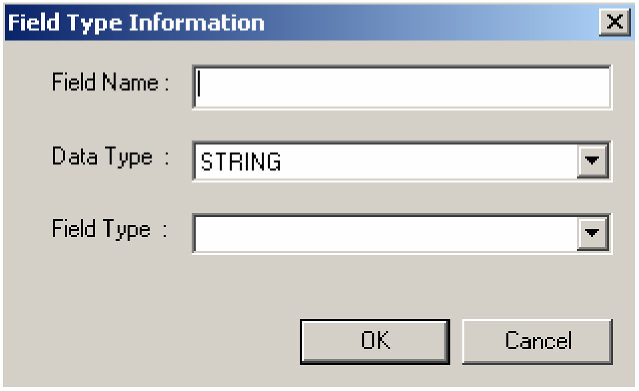
Pattern: 703 605-6900

Start Row: 6 Start Column: 18

End Row: 7 End Column: 31

NOTE: Field length is used to set the database field length for indexing. An incorrect value may cause problems during indexing.

6 Click **New**. DefTool displays the Field Type Information dialog. Specify a name for the subkey field, select the data type, and select **Sub key** from the Field Type drop-down list.



Data Type : STRING

7 Click **OK**. Continue to define the subkey group field extraction rules (properties) including the page style to associate with the group field, and click **OK** when you are done. DefTool displays the new subkey group field under the group and its associated markers in the Definition Tree; note that the subkey group field icon looks different from the regular group field icon.

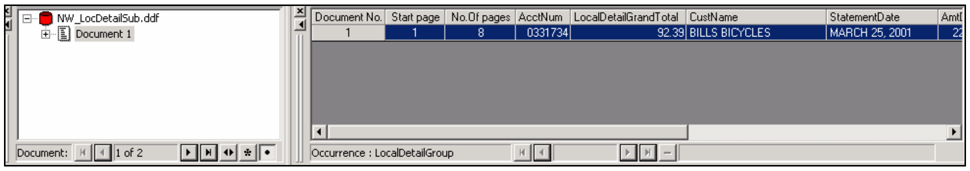
8 On the Group Properties dialog, define any additional group fields you want within the sub-account parent group. You can also define group fields in nested groups. You can define more than one group field for a group, but you must define only one as the subkey.

9 Use the DefTool Simulator, described below, to test extraction of subkeys and group table data from your data input file. To edit data extraction rules, right-click a group or on a marker in the Definition Tree and select **`Edit`** from the right-click menu. To edit a subkey or other group field extraction rules, see "Modifying a Group Field or Subkey Properties."

10 Save the DDF.

To simulate extraction of sub-account group data and subkeys in DefTool:

1 Click the Extractor Simulation icon in the Simulation toolbar or select **`Simulator>Extractor Simulation`**. The Simulation pane appears at the bottom of the DefTool main window. Nodes shown in the tree on the left represent each document detected in the input file. On the right, Simulator displays the document boundaries and extracted fields.

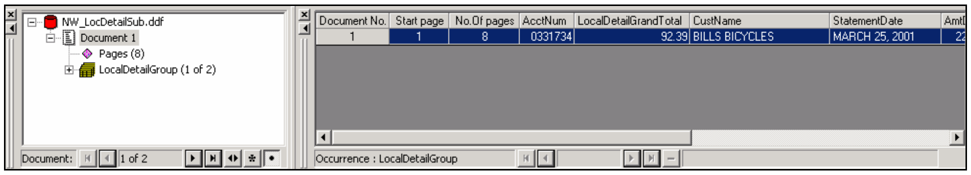


NW_LocDetailSub.ddf
Document 1

Document No.	Start page	No.Of pages	AcctNum	LocalDetailGrandTotal	CustName	StatementDate	AmtI
1	1	8	0331734	92.39	BILLS BICYCLES	MARCH 25, 2001	22

Document: 1 of 2
Occurrence : LocalDetailGroup

2 Click the document to expand the list of pages, groups, and fields.

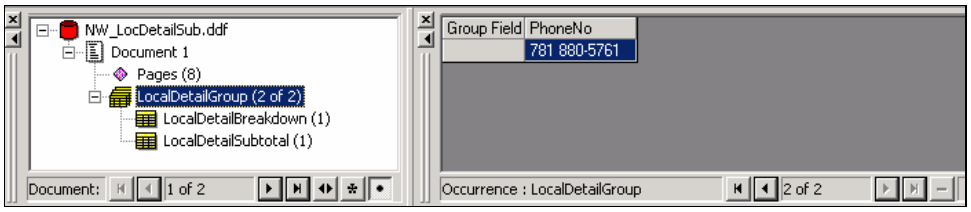


NW_LocDetailSub.ddf
Document 1
Pages (8)
LocalDetailGroup (1 of 2)

Document No.	Start page	No.Of pages	AcctNum	LocalDetailGrandTotal	CustName	StatementDate	AmtI
1	1	8	0331734	92.39	BILLS BICYCLES	MARCH 25, 2001	22

Document: 1 of 2
Occurrence : LocalDetailGroup

3 Click the sub-document group to display the subkey or group fields for the first occurrence.

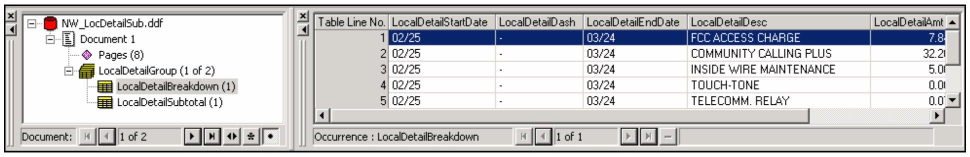


NW_LocDetailSub.ddf
Document 1
Pages (8)
LocalDetailGroup (2 of 2)
LocalDetailBreakdown (1)
LocalDetailSubtotal (1)

Group Field	PhoneNo
	781 880-5761

Document: 1 of 2
Occurrence : LocalDetailGroup 2 of 2

4 To display group data extraction, click the table.



NW_LocDetailSub.ddf
Document 1
Pages (8)
LocalDetailGroup (1 of 2)
LocalDetailBreakdown (1)
LocalDetailSubtotal (1)

Table Line No.	LocalDetailStartDate	LocalDetailDash	LocalDetailEndDate	LocalDetailDesc	LocalDetailAmt
1	02/25	-	03/24	FCC ACCESS CHARGE	7.8
2	02/25	-	03/24	COMMUNITY CALLING PLUS	32.2
3	02/25	-	03/24	INSIDE WIRE MAINTENANCE	5.0
4	02/25	-	03/24	TOUCH-TONE	0.0
5	02/25	-	03/24	TELECOMM. RELAY	0.0

Document: 1 of 2
Occurrence : LocalDetailBreakdown 1 of 1

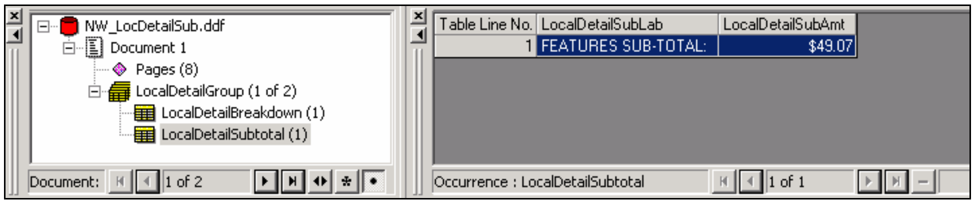
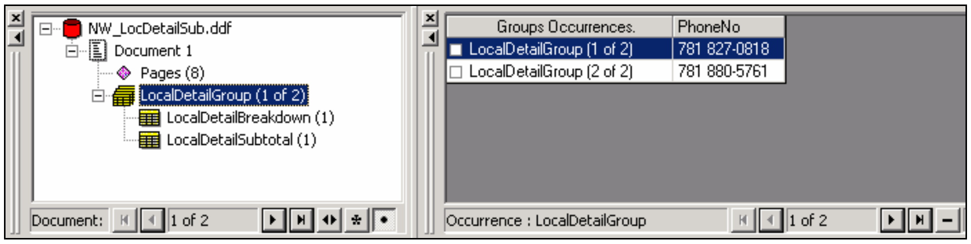


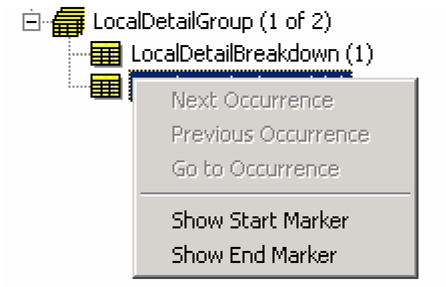
Table Line No.	LocalDetailSubLab	LocalDetailSubAmt
1	FEATURES SUB-TOTAL:	$49.07

5 To test the extraction of all subkeys, click the Show All Tables icon, click the Document to expand the list of fields, tables, and groups, then click the sub-document data group. DefTool displays the extracted subkey and group fields for each group occurrence in the document.



Groups Occurrences.	PhoneNo
LocalDetailGroup (1 of 2)	781 827-0818
LocalDetailGroup (2 of 2)	781 880-5761

6 You can use all the existing features of the DefTool Simulator to test the groups, tables, and markers. For example, you can right-click on a group or table to display a menu of options:



7 As with any groups, tables, or fields, you can choose to selectively simulate a sub-document group (select **Simulator>Simulator Settings** and the Group tab or click the Simulator Settings icon). By default, DefTool Simulates all tables, group fields, and subkeys associated with the group.

Adding a Group Field

To add a group field:

1 With the DDF open in DefTool, right-click on the group name in the Definition Tree and select **Edit** from the right-click menu.

2 On the Group Properties dialog, click **Create**. Use the left mouse button to drag the crosshair selector box around the data you want to use for the group field. (Define the group fields within the sub-account parent group.)

3 At the Select Your Choice dialog, select **Field** and click **OK**. DefTool displays the Group Field Properties dialog.

4 Define the field and the extraction rules as described in “Defining Sub-Account Groups, Subkey, and Group Fields” above.

Modifying a Group Field or Subkey Properties

You can edit the properties of a group field, including a subkey, after you have created it. For example, the simulation process may reveal a problem that requires you to edit the group field extraction rules, such as the field pattern or window coordinates.

To edit the properties of a group field:

1. With the DDF open in DefTool, expand the Definition Tree to expose the group field.
2. Right-click on the group field (can be a subkey), and select **Edit** from the right-click menu. The Group Field Properties Dialog appears.
3. Make your edits using the Group Field Properties Dialog and click **OK**.

Deleting a Group Field or Subkey

You can delete a group field, including subkeys, from a DDF. If a group field or subkey has multiple locations, you can select and delete by location (similar to how fields work).

To delete a group field or subkey:

1. With the DDF open in DefTool, expand the Definition Tree to expose the group field.
2. Right-click on the group field (can be a subkey), and select Delete from the right-click menu. DefTool asks if you’re sure you want to delete the field.
3. Click **Yes**. DefTool deletes the group field.

To delete multiple group fields:

1. With the DDF open in DefTool, expand the Definition Tree to expose the group.
2. Right-click the group and select **Edit** from the right-click menu. DefTool displays the Group Properties dialog.
3. Under the Included Group Fields, select a field to delete and click **Delete**. DefTool asks if you’re sure you want to delete the field.
4. Click **Yes** to delete the field. DefTool deletes the group field.
5. Continue deleting group fields as necessary.
6. Click **OK**.

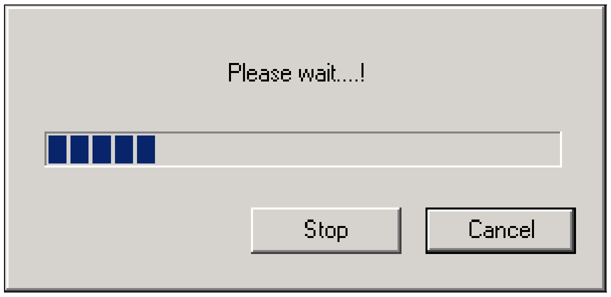
11 Using the Extraction Simulator

Using Simulator for the First Time

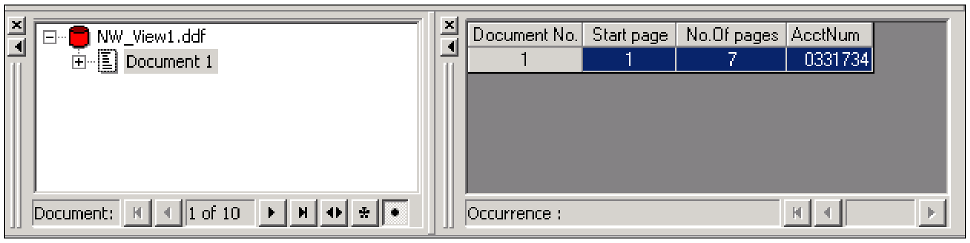
The Extraction Simulator utility is built into the DefTool to allow you to test your DDF extraction rules as you are defining them, without having to go through the process of indexing the data source by the production components. This utility prevents the "blind design" problems often associated with application development. After you have defined the Document Style and Primary Key Field, you can use the Extraction Simulator at any time while you are working in the DefTool.

By default, the Extraction Simulator will test all of the definitions you have made simultaneously, every time you run it. You do not need to make any configurations to use the Extraction Simulator for complete simulation.

- Locate the **`Simulator`** icon in the Simulation Tool Bar or select **`Simulator>Extraction Simulator`**. (Click the **`Extractor Simulation`** Icon to test your Document Style and Primary Key field definitions.) A Please Wait...! prompt appears (shown below):



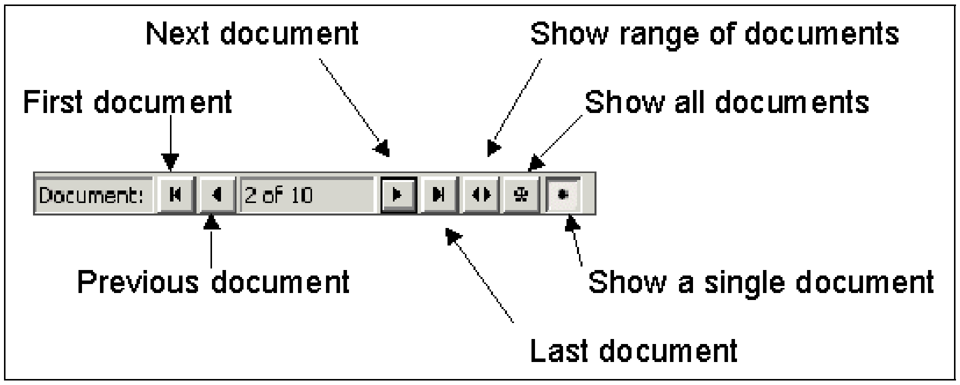
A second horizontal pane will appear in the DefTool main console. On the left side, the Tree contains nodes representing each statement (referred to as documents here) detected in the data source, and on the right side, a table displays the document number, start page, number of pages and fields extracted for each statement detected in the data source (shown below).



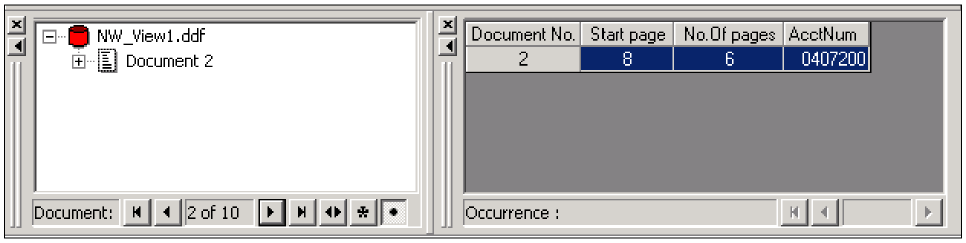
Document No.	Start page	No.Of pages	AcctNum
1	1	7	0331734

Navigating in the Simulator

By default, the Simulator view only displays the first statement detected. The Simulation View contains a Navigational Bar, located below the Simulation Tree (shown below). The navigational tools can modify the Simulation view to show a subsequent statement, include a range of statements, or display all statements in the data source.

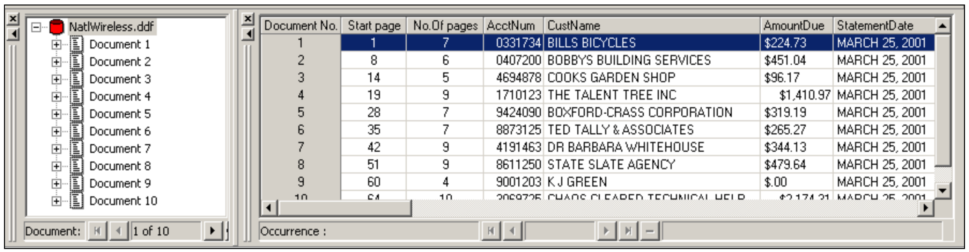


- Click the **`Next Document`** ▸ icon to verify the detection of Statement 2 and its Primary Key field extraction (0407200). The Tree View displays a node for Document 2, and the Simulation View displays the document number, start page, number of pages and (AcctNum) Primary Key field extraction.



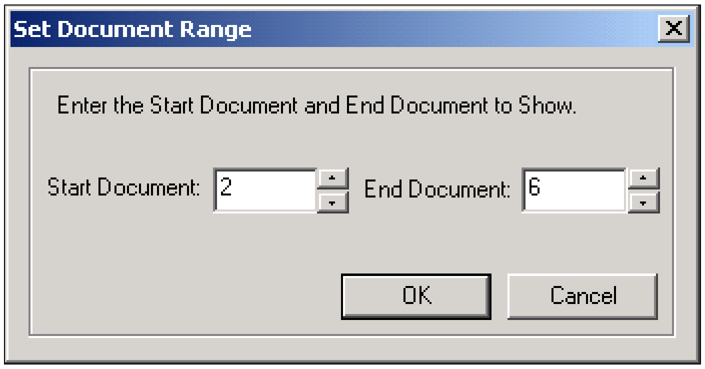
Document No.	Start page	No.Of pages	AcctNum
2	8	6	0407200

- Click the **`Show All Documents`** ✱ icon. If you have properly defined your Document Style and Primary Key field, all statements in the data source will be represented in both the Tree View and Simulation View (shown below).



Document No.	Start page	No.Of pages	AcctNum	CustName	AmountDue	StatementDate
1	1	7	0331734	BILLS BICYCLES	$224.73	MARCH 25, 2001
2	8	6	0407200	BOBBYS BUILDING SERVICES	$451.04	MARCH 25, 2001
3	14	5	4694878	COOKS GARDEN SHOP	$96.17	MARCH 25, 2001
4	19	9	1710123	THE TALENT TREE INC	$1,410.97	MARCH 25, 2001
5	28	7	9424090	BOXFORD-CRASS CORPORATION	$319.19	MARCH 25, 2001
6	35	7	8873125	TED TALLY & ASSOCIATES	$265.27	MARCH 25, 2001
7	42	9	4191463	DR BARBARA WHITEHOUSE	$344.13	MARCH 25, 2001
8	51	9	8611250	STATE SLATE AGENCY	$479.64	MARCH 25, 2001
9	60	4	9001203	K J GREEN	$.00	MARCH 25, 2001
10	64	10	2069725	CHAOS CLEARED TECHNICAL HELP	$2,174.21	MARCH 25, 2001

- Click the **`Show Range of Documents`** ◂▸ icon. The Set Document Range dialog appears. Adjust the start and end documents as desired (shown below). Since this is a live dialog, you need to set the end document first. The Simulation View will display the extraction for the document range you specified.


Set Document Range

Enter the Start Document and End Document to Show.

Start Document: 2

End Document: 6

At this point in the DDF creation process, the Simulator is useful for determining if the statements will be detected. The Document Style and Primary Key field are used together to detect a statement in your data sources. If both of these settings are not properly defined, and therefore not detecting particular statements, then you will notice this immediately by the absence of the statements in the Simulation View. The Simulation View would consider these statements as part of the previous statements. This is similar to what would happen in a production environment. If your Simulation View doesn't appear as expected, you will need to edit your definitions and rerun the Simulator.

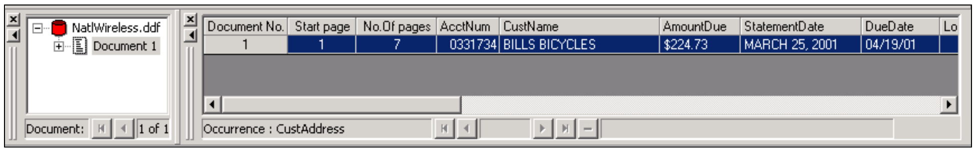
Information in the Simulation View is only as current as the last time you ran Extractor Simulation.

Simulating Field Extractions

The Extraction Simulator will allow you to test your rules for fields, tables and groups before you deploy the application. You can perform complete or selective extraction simulation on any of the rules you have created.

By default, the Extraction Simulator will test all of the rules (fields, tables and groups) that you have built in your DDF. All the fields that you have defined will appear in the main Simulator View after you have run the Simulator.

Click the **`Simulator`** icon to test your new field definitions. After the Please Wait...! window appears and closes, your DefTool main console will include the Simulator Data View (right-pane) and Tree (left-pane), both shown below.



Document No.	Start page	No.Of pages	AcctNum	CustName	AmountDue	StatementDate	DueDate	Lo
1	1	7	0331734	BILLS BICYCLES	$224.73	MARCH 25, 2001	04/19/01	

Occurrence : CustAddress

The default Simulator View appears displaying the fields of the first statement (or document, as called here) in your sample data source. Follow the instructions below to modify the View of your simulation.

To See This…	Do This…
View the fields of document 1	Scroll to the right in the Simulator Data View
View the fields of document 2	Click the Next Document icon in the left-pane
View the fields of all documents	Click the Show All Documents tool in the left-pane. Scroll to the right in the Simulator Data View
Verify that your fields are extracting the right data	Use the Simulator Data View and the chart of fields below.
Sort each field in descending order.	Click once on each field name, one at a time, in the Simulator Data View.
Sort each field in ascending order.	Click twice on each field name, one at a time, in the Simulator Data View.
Troubleshoot your field definitions	Look for “Hit Not Found” where there should be field occurrences. Double click “Hit Not Found” instances in the Simulator View to navigate to the corresponding area in the data source.

Simulating Table Extractions

The Extraction Simulator allows you to test your rules for fields, tables and groups before deploying the application. You can perform complete or selective extraction simulation on any of the rules you have created.

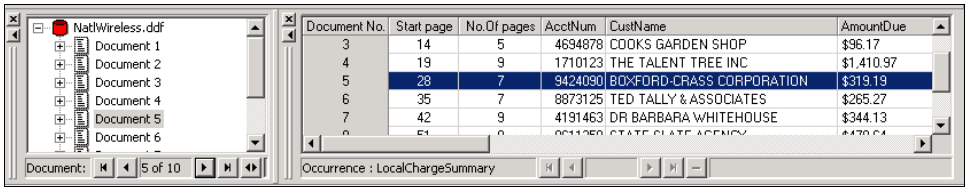
To test new table definitions:

- Click the **`Simulator`** icons. Your Simulation Data View appears as it has after previous extractions, with all fields presented in the main view area.

By default, the Extraction Simulator tests all the rules (fields, tables and groups) you have built in your DDF. DefTool accesses the extraction results for your table definitions from the Simulation Tree, one document at a time.

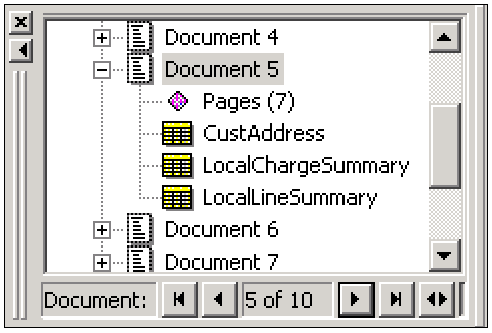
To view the table extraction results for any document (or statement):

1. Locate the document in the Simulation Tree — enable Show All Documents, Show Range or Show Single Document, if necessary, to view the desired document in the Tree. (Enable Show All Documents to locate Document 5 in the Simulation Tree as shown below.)



Document No.	Start page	No.Of pages	AcctNum	CustName	AmountDue
3	14	5	4694878	COOKS GARDEN SHOP	$96.17
4	19	9	1710123	THE TALENT TREE INC	$1,410.97
5	28	7	9424090	BOXFORD-CRASS CORPORATION	$319.19
6	35	7	8873125	TED TALLY & ASSOCIATES	$265.27
7	42	9	4191463	DR BARBARA WHITEHOUSE	$344.13

2 Expand the tree node for the document by clicking on the corresponding plus box in the Simulation Tree. (Expand Document 5 in the Simulation Tree as shown below.) Each table that was detected and extracted from the document will be listed as sub-nodes.



3 To view the extraction results of a table for this specific document, click the desired table in the now-expanded Simulation Tree. (In the Simulation Tree, click **`LocalChargeSummary`** to view the extraction of the Local Charge Summary data in the *BOXFORD-CRASS* statement, or Document 5 as shown below.) The data extracted as your table will be shown in the Simulation Data View.

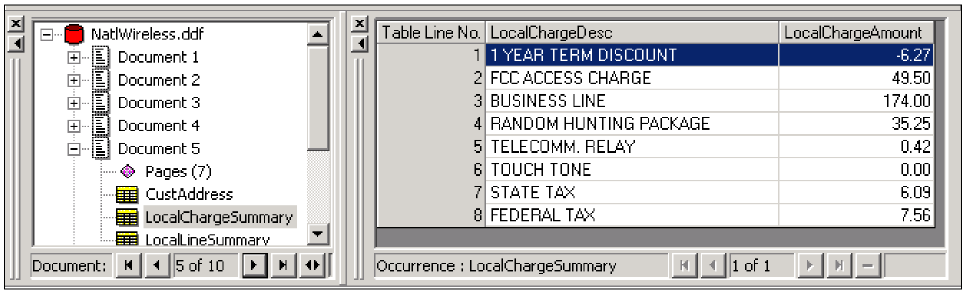


Table Line No.	LocalChargeDesc	LocalChargeAmount
1	1 YEAR TERM DISCOUNT	-6.27
2	FCC ACCESS CHARGE	49.50
3	BUSINESS LINE	174.00
4	RANDOM HUNTING PACKAGE	35.25
5	TELECOMM. RELAY	0.42
6	TOUCH TONE	0.00
7	STATE TAX	6.09
8	FEDERAL TAX	7.56

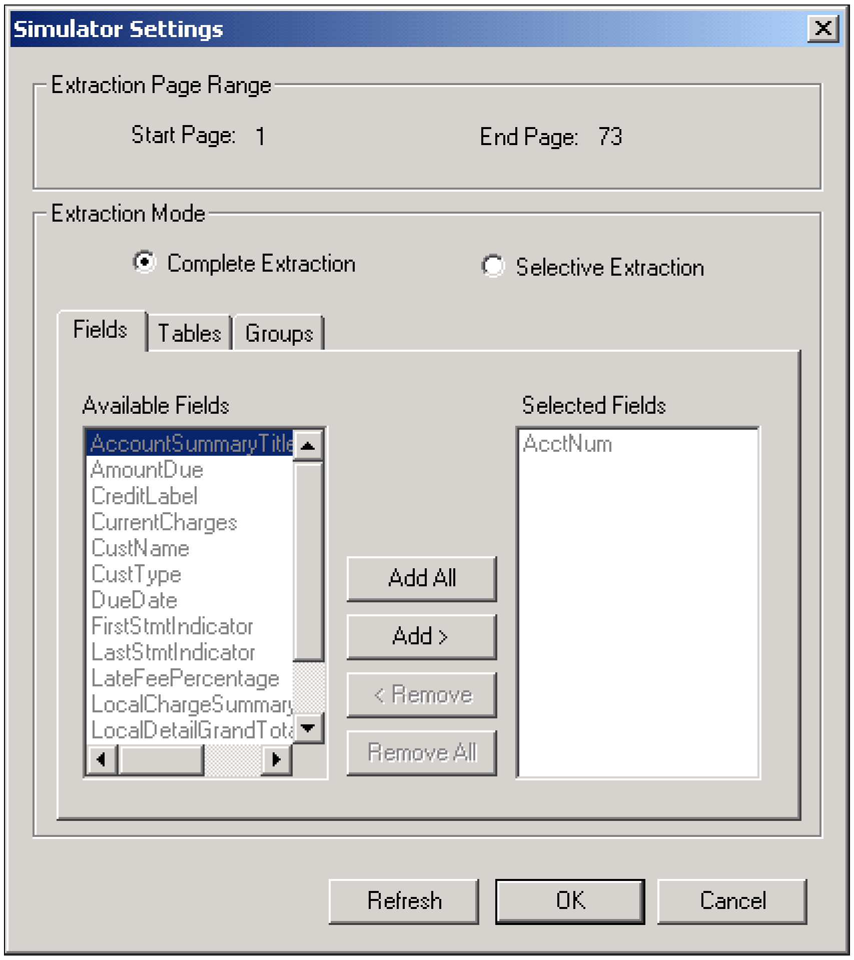
(Compare the results in your Simulation Data View with the print-copy and DefTool work area view of the *BOXFORD-CRASS* statement, and with the image above.)

Simulating Selective Extraction

You must configure the Extraction Simulator to test the rules for a selection of fields, tables and groups within the Simulator Settings dialog.

To configure Simulator Settings:

1 Click the Simulator Settings icon in the Simulation Tool Bar or select **`Simulator>Simulator Settings`**. The Simulator Settings dialog opens (shown below).



2 By default, the Simulator is set to perform extraction with all of the rules you have defined. To override the Complete Extraction and select only the items you want, click the **`Selective Extraction`** radio button. Items in the Fields, Tables and Groups tabs will become active.

3 To select the fields you want extracted, go to the Fields tab, Available Fields window. For each desired field:

- Click the field name, and click **Add…**, or
- Double-click the field name.

(Select the following fields for extraction so that they appear in the Selected Fields window: **`CustName`**, **`AmountDue`**, and **`CustType`**.)

4 To select the tables you want extracted, go to the Tables tab, Available Tables window, and for each desired table:

- Click the table name, and click **Add** or
- Double-click the table name.

(Select the following tables for extraction so that they appear in the Selected Tables window: **`LocalChargeSummary`** and **`LocalLineSummary`**.)

5. To select the groups you want extracted, go to the Groups tab, Available Groups window, and for each desired group:
   - Click the group name, and click **`Add,`** or
   - Double-click the group name.
6. Click **`OK`** to save your settings and close the Simulator Settings dialog. (Click **`OK`**.)
7. Click the **`Simulator`** icon to re-simulate your newly selected extraction rules.

   TIP: **TIP:** You can later re-configure your Simulator Settings within the same dialog, by selecting fields, tables and groups, respectively, and clicking **`Add >`**, **`Add All`**, **`Remove >`** or **`Remove All`**. You can delete your **`Selective Extraction`** settings quickly within the same dialog by clicking on **`Refresh`**

12 Regular Expressions

What is a Regular Expression

A regular expression is a pattern that you use in a DDF rule, to validate a data string that you want to extract. The data string that your rule is looking for is usually of a specified format. You use regular expressions in all of your DDF rules to establish the data pattern that is to be looked for, whether the data that your rule is looking for is static or variable. In order to write regular expressions that are effective in extracting all the data values you want in your statements, and allow for optimal extraction performance, a good understanding of regular expression syntax is necessary.

Purpose of Regular Expressions

To review, a pattern in any DDF rule can:

- Validate the data string that identifies your document style or additional page styles
- Verify the start and end marker positions of tables and groups in your statements
- Extract the appropriate data values for fields and table columns.

A regular expression in a DDF rule tells the eStatement Manager extraction components, position by position, exactly what characters are valid as part of the data value.

Every DDF contains rules to extract data values that conform to a well-known format. Some common data formats that will likely appear in your data (and that you will want to extract) are:

- Dates
- Currency
- Phone Numbers

City, State and Zip Code Numeric values Statement data will usually fit a proprietary pattern, such as:

- Account Numbers
- Customer ID Numbers

Finally, you will likely extract some static text, or at least use static data strings as markers and page style identifiers. A regular expression is needed to find the data strings during live statement retrieval. These strings can be:

- Descriptive labels, such as TOTAL AMOUNT DUE:
- Print or database file control codes
- Headings, such as LOCAL DETAIL CHARGES

To extract the data that fit common formats like the ones listed above, you can write a regular expression that will uniquely identify a data value and exclude other data values that do not follow the format.

Using Regular Expressions with eStatement Manager

When your customers click to view their dynamically composed statement, the eStatement Manager extraction components use the regular expression pattern that you define to validate the data value found in a left-to-right, one-ahead manner. This means that if the regular expression does not match a position of the data value found where it is expected, it will proceed to the next position in the regular expression in its attempt to validate the position. If the data value found does not then match the next position in the regular expression, the data will not be extracted.

Writing Regular Expression Patterns

The illustrations below will introduce you to the concept of writing a regular expression to detect and extract data values of specific formats with your DDF rules. Each illustration contains

- A sample of data that follow a common format
- An examination of the data sample to find the general pattern that exists
- A regular expression that will find and extract all data that fit the pattern that was found
- An explanation of each new regular expression operator used.

You will become acquainted with all of the most commonly used regular expression operators through the succeeding illustrations.

Illustration 1: Shorthand Date Formats MM/DD/YYYY

Consider the field DueDate, whose representative sample of values across statements are:

- 03/16/1997
- 12/31/2000
- 10/05/1999

To write a regular expression to extract these values as DueDate, you start by finding out what the different examples have in common. Given that this sample includes all possibilities of DueDate values, you can see that the general pattern is (per position):

Pattern position	Used for	Valid characters
Position 1	The first digit of the MONTH	0 or 1
Position 2	The second digit of the MONTH	Any number
Position 3	Delimiter of MONTH and DAY	A forward slash
Position 4	The first digit of the DAY	0, 1, 2 or 3
Position 5	The second digit of the DAY	Any number
Position 6	Delimiter of DAY and YEAR	A forward slash
Position 7	The first digit of the YEAR	1 or 2
Position 8	The second digit of the YEAR	Any number
Position 9	The third digit of the YEAR	Any number
Position 10	The fourth digit of the YEAR	Any number

Therefore, using regular expression syntax, your regular expression pattern to validate and extract values of the DueDate would be:

```
[01][0-9]/[0-3][0-9]/[12][0-9][0-9][0-9]
```

To specify that multiple characters are valid for a position in the data value, enclose the characters within hard brackets (examples shown below):

Regular Expression	Signifies
`[01]`	`0 or 1`
`[AF]`	`A or F`
`[Aa]`	`A or a`
`[az09]`	a, z, 0 or 9

To specify a range of characters that are valid for a position in the data value, separate the first and last character in the range with a dash (examples follow):

Regular Expression	Signifies
`[0-3]`	0, 1, 2 or 3
`[0-9]`	Any number
`[A-F]`	A, B, C, D, E or F
`[A-Z]`	Any uppercase letter
`[a-z]`	Any lowercase letter

TIP: To specify that only ONE character is valid for a position in the data value, simply enter that character into the pattern. In the example above, Positions 3 and 6 must be a forward slash, therefore the regular expression contains a ***/*** in each of these positions.

Illustration 2: Varied Shorthand Date Formats

Consider what would happen if DueDate values could also be:

- 3/25/1998
- 12/1/2000
- 05-07-1996

This changes the general pattern of the DueDate values.

Pattern position	**Used for**	**Valid characters**	**Appearance frequency**
Position 1	The first digit of the MONTH	0 or 1	Optional
Position 2	The second digit of the MONTH	Any number	Exactly once
Position 3	Delimiter of MONTH and DAY	A forward slash or a dash	Exactly once
Position 4	The first digit of the DAY	0, 1, 2 or 3	Optional
Position 5	The second digit of the DAY	Any number	Exactly once
Position 6	Delimiter of DAY and YEAR	A forward slash or a dash	Exactly once
Position 7	The first digit of the YEAR	1 or 2	Exactly once
Position 8	The second digit of the YEAR	Any number	Exactly once
Position 9	The third digit of the YEAR	Any number	Exactly once
Position 10	The fourth digit of the YEAR	Any number	Exactly once

Therefore, your regular expression would change to allow these varying formats in the DueDate:

```
[01]*[0-9][/\-][01]*[0-9][/\-][12][0-9][0-9][0-9]
```

To specify that a character could appear multiple times, or not appear at all (i.e. 0 or more times), enter an asterisk after the character. To specify that a selection of characters could appear zero or more times, enter the asterisk after the closing hard bracket that contains the characters (examples follow).

Regular Expression:	**What it means:**
[0-9]*	A numeric character is optional, and could occur more than once (no limit)
[0-9,]*	If a character appears, it must be a numeric or a comma, and there could be multiple occurrences of any of these characters

To extract a regular expression syntax character as normal data, you must enter a backslash before that character, whether it appears as part of a selection of characters for a position (i.e. within a set of hard brackets) or in its own position (examples follow).

Regular Expression	**Signifies**
\-	The dash is the only character that can and must appear in the position
[/\-]	The valid characters for this position are the forward slash and dash
[\-\+*]	The valid characters for this position are the dash, plus and asterisk
No\.	Look for the data string *No.* (i.e. Position 1 = N, Position 2 = o, Position 3 =.)

Illustration 3: Currency Values

Consider a field, AmountDue, whose representative sample of values across statements are:

- $964.23
- $1.99
- $65.00
- $0.03

Again, to write a regular expression to extract these values as AmountDue, you start by finding out what the different examples have in common. It is not necessary, however, to try and decipher the pattern from left to right at this point. Simply look for the common traits of the sample data, and build your regular expression around those traits. Given that this sample includes all possibilities of AmountDue values, you can see that the general pattern is (per position):

Pattern position	**Used for**	**Valid characters**
First Position	The currency symbol	$
Next Position(s)	The dollar amount	Any number, one or more times
Next Position	The decimal point	.
Next Position	The 10th of a cent figure	Any number
Last Position	The 100th of a cent figure	Any number

Therefore, using regular expression syntax, the regular expression pattern to extract values that match the above pattern would be:

```
$[0-9]+\.[0-9][0-9]
```

Currency can be sorted to three decimal places. For further information, see Appendix A – Data Definitions.

To specify that a character could appear 1 or more times (i.e. always occurs at least once, but may occur in subsequent positions), enter a plus symbol after the character in the regular expression. To specify that a selection of characters could appear 1 or more times, enter the plus after the closing hard bracket that contains the characters (examples shown below).

Regular Expression	**Signifies**
[0-9]+	At least one numeric character must appear, but there could be more than one (no limit)
A+	At least one A must appear, but there could be more
[A-Za-z0-9]+	Any alpha-numeric character is valid for one or more positions in a data value
+ (space, then +)	One or more spaces are required in the data string

Illustration 4: Longhand Date Formats

In the StatementDate field, a representative sample of values across statements are:

- May 25, 2001
- August 3, 1998
- January 20, 1999

Again, to write a regular expression to extract these values as StatementDate, determine what the different examples have in common and build your regular expression as specifically as possible, given those common traits. Given that this sample includes all possibilities of StatementDate values, you can see that the general pattern is:

Pattern position	**Used for**	**Valid characters**
First Positions	The longhand month of the year	January, February, ..., and December
Next Position	A space between the month and day	A space character
Next Position(s)	The 1- or 2-digit numeric day of the month	One or two numbers
Next Position	A comma after the day	A comma
Next Position	A space between the comma and year	A space character
Last 4 Positions	The numeric 4-digit year	[12][0-9][0-9][0-9]

Therefore, using regular expression syntax, the regular expression pattern to extract values that match the above pattern would be:

```
(January|February|March|April|May|June|July|August|September|O
ctober|November|December) [0-3]*[0-9], [12][0-9][0-9][0-9]
```

To specify that a selection of data strings are valid for detection or extraction by your DDF rule, delimit the strings with the pipe symbol and enclose the entire string set with soft brackets. The data strings you specify can be static (as in the case of longhand months, above) or variable (i.e. using an inner regular expression pattern) (examples shown below).

Regular Expression	**Signifies**
`(cat|dog)`	The string cat, or the string dog
`([A-Z][a-z]+|[0-9]+)`	An alpha string starting with an uppercase letter, e.g. Apple, or a string of numbers, e.g. 01760
`([a-z0-9.\-]+@[a-z.\-]+|\([0-9][0-9][0-9]\)[0-9][0-9][0-9]\-[0-9][0-9][0-9][0-9])`	An email address, e.g. biller-2@bbb.com, or a phone number, e.g. (508) 652-8600

Illustration 5: Data Values that Vary Widely in Pattern

Consider a table column called AddressLine that is defined to extract each line of a customer address. Examine the given representative sample of address lines below:

- 2 Apple Hill
- # 4-B
- Three Canal Park
- c/o Libby Mae Brown

- % John Talley

These potential values of the AddressLine table column vary widely, both in character composition and in length.

Therefore, it can be difficult, to find a common pattern among a data sample like above.

A regular expression to extract each position of each AddressLine value might look like:

```
[A-Za-z0-9#%][A-Za-z0-9#%\-\(\)\.&@,/:’” ]+
```

This regular expression requires that:

- Position 1 must be an alpha-numeric character, a pound sign or a percent sign
- Positions 2 and beyond can each be an alphanumeric, a pound, a percent, a dash, etc.

This regular expression is very broad, so that it must check many characters per position. This practice is:

- Inefficient for extraction performance, and possibly incomplete, given that there may be characters that appear in AddressLine that you have not anticipated.

You can write a regular expression that specifies which characters are NOT valid for a position or positions. This will:

- Exclude unwanted data strings that exist within the search area, and
- Extract the correct data string(s) as desired.

A regular expression that would extract all Address Lines, yet exclude space lines that may appear between rows of meaningful data (i.e. no space in the first position), is shown below:

```
^ .+
```

Pattern position	**Used for**
Position 1	All characters are valid except for a space
Position 2 and on.	All characters are valid

Illustration 6: Static data strings

Suppose that you want to build a DDF rule that needs to locate (or extract) a static data string. For example, you will likely use static strings as your anchors, page styles, and table/group markers. Sometimes, you will even want to extract static strings that are related to some variable data, so that you can dynamically present that information when its related data appears in the statement, rather than hard-code it into your HTML templates.

Points to keep in mind when you use a static string as the basis for an anchor, marker, page style, or any other rule:

- The pattern you write will specify exactly ONE character per position, rather than a range or selection of characters

- If only one character is valid for a position, there is no need to surround it with brackets or place repeat operators (such as + or *) after it (unless you expect it to repeat, of course!)
- The pattern is, however, technically still a regular expression, even though it may not use regular expression operators.

Suppose you want to use the string DETAIL as a marker for a table or group? Given that you expect the string to be exactly the same across all statements, the pattern to find this string is:

Pattern position	**Used for**
Position 1	An uppercase D only
Position 2	An uppercase E only
Position 3	An uppercase T only
Position 4	An uppercase A only
Position 5	An uppercase I only
Position 6	An uppercase L only

It is easy to see that the regular expression to find the data string DETAIL is:

```
DETAIL
```

Illustration 7: Partially Static Data Strings

Sometimes, only a portion of the data string you are looking for is static. Then, your pattern will be partially static as well.

Points to keep in mind when you use a partially static string as the basis for an anchor, marker, page style, or any other rule:

- Part of the pattern you write will specify exactly ONE character per position, whereas the other portion of the pattern will be variable.
- If only one character is valid for a position, there is no need to surround it with brackets or place repeat operators (such as + or *) after it (unless you expect it to repeat, of course!)

Suppose you want to extract the DeptNumber as a field, including the descriptive text:

- DEPT 1
- DEPT 2
- DEPT 10

Given that the sample above is broad enough to represent all values of DeptNumber in the statements, you can see that the pattern is:

Pattern position	**Used for**
Position 1	An uppercase D only
Position 2	An uppercase E only
Position 3	An uppercase P only
Position 4	An uppercase T only
Position 5	A space only
Position 6	A number
Position 7 (optional)	Another number, if anything

Therefore, using regular expression syntax, the regular expression pattern to extract values that match the above pattern would be:

```
DEPT [0-9]+
```

Regular Expressions Reference Chart

The chart below is intended as a job aid for you to use during the development of your DDFs.

Regular expression	**Used for**	**How it is used in a regular expression**	**How it is NOT used in a regular expression**

[ ]	Contains a set of characters that are valid for a position
	Can be used with -
		Surrounds a set of 2 or more characters that are valid
		[01]
		Used with - , surrounds a range of characters that are valid (used with the dash - ; see below):
		[a-f]
		Used with - , surrounds both single characters and a range that are valid (used with the dash - ; see below):
		[0-9,]
			NOT used to contain data strings:
			Bad = [January|February]
			Good = (January|February)
			NOT used in a nested fashion:
			Bad = [[xyz]0-9]
			Good = [xyz0-9]
			NOT used for one character that is valid:
			Bad = [$]
			Good = $

Regular expression	**Used for**	**How it is used in a regular expression**	**How it is NOT used in a regular expression**

-	Delimits a range of characters
	Always used within [ ]
		Always used within [ ], separates the first and last character of a range that is valid
		[5-9]
		Can be used multiple times per position:
		[A-Za-z0-9]
			NOT used to state two characters that are valid:
			Bad = [0-1]
			Good = [01]
			NOT used without a first or last character:
			Bad = [A-]
			Good = [A-Z]

+	One or more of the preceding character(s) are valid	After a single character that is valid:
		0+
		Used outside [ ], after a set of characters that are valid:
		[0-9]+
			NOT used within hard brackets:
			Bad = [0-9+] (will mean numbers and a + are valid)
			Good = [0-9]+
			NOT used before a character or selection of characters:
			Bad = + (preceding a space)
			Good = + (i.e. one or more spaces are valid)
			Bad = +[a-z]
			Good = [a-z]+

*	"Optional"
	Zero or more of the preceding characters are valid
		After a single character that is valid:
		0*
		Used outside [ ], after a set of characters that are valid:
		[0-9,]*
		Will be treated as normal character, not operators, if it appears inside square brackets:
		[0-9+]
			NOT used within hard brackets:
			Bad = [0-9*]
			Good = [0-9]*
			NOT used before a character:
			Bad = *$
			Good = $*
			NOT used before a selection of characters:
			Bad = +[a-z]
			Good = [a-z]+
			NOT used consecutively (will yield unpredictable results):
			Bad = [0-9]*,*[0-9]*\.*[0-9]*[0- 9]*
			Good = [0-9,]*\.[0-9]*

.	"Wildcard" Any character found is valid	Used by itself in one position: . Used to extract widely varying strings of unknown length: .+	NOT used within hard brackets as part of a selection of valid characters: Bad = [a-f.] (ambiguous)

Regular expression	**Used for**	**How it is used in a regular expression**	**How it is NOT used in a regular expression**

( )	Surrounds a selection of data strings that are valid	Used with | , surround the data strings that are valid:
		(Total Amt Due|Final Amt Due)
			NOT used within [ ]
			Bad = [(cat|dog)A-Z]
			Good = (cat|dog|[A-Z])

|	Delimit a selection of data strings that are valid
	Always used within ( )
		Used with ( ), in between each valid data string:
		(MA|CT|RI|VT|NH|ME)
			NOT used without ( ):
			Bad = Local|Long Distance
			Good = (Local|Long Distance)

\	Extract the following regular expression operator as a normal character
	Used with W,w, N or n, extract appropriate set of characters (see below)
		Used within [ ] or ( ), precedes the regular expression operator character you want to extract:
		(Jan\.|Feb\.|Mar\.)
		[0-9,\.]
		Used in a static position, precedes the regular expression operator you want to extract:
		No\. of Shares
		Used to extract itself:
		\\
			NOT used after a regular expression operator you want to extract:
			Bad = +\
			Good = \+

^	The following character is NOT valid	Used before [ ], to signify that the characters contained within are not valid:
		^[a-z]
		Used before ( ), to signify that the data strings contained within are not valid:
		^(Dept|Section|group)
		Used before a single character:
		^ (I.e. a space is not valid)
			NOT used after a character:
			Bad = $^
			Good = ^$

\w	Extract any alpha character (upper- or lowercase)	Used within [ ], if part of a bigger set of characters that are valid: [01\w] Used alone in a position: \w	NOT used within ( )
\W	Extract any NON-alpha character	(See above)	(See above)
\n	Extract any numeric character	(See above)	(See above)
\N	Extract any NON-numeric character	(See above)	(See above)

13 Dynamic Pattern Matching

What is a Dynamic Field?

A dynamic field is a parameter that you create in your DDF, for use in the pattern of a group marker, table marker, or table column, to allow for the selective extraction of the group or table data. When a statement is retrieved dynamically in an application using dynamic pattern matching, the dynamic field parameter in the group marker, table marker, or table column is replaced with a static string of the recipient's choice during live retrieval, and only the data occurrences whose marker or column match the string get extracted.

Dynamic pattern matching allows you to build an application that will extract and present data hierarchically, and specific to the recipient's requirements. For example, you may want your recipient to be able to view their statement in "pieces," rather than all at once. The recipient's first "piece," or view, of their statement, may show top-level summary data only, from which he or she can select the portion of detail to be viewed dynamically, based on a selection in the Summary view. This functionality allows for faster extraction and presentment performance, and also facilitates easier viewing for recipients, in that they would no longer be forced to scroll through potentially thousands of lines of data in their browser.

With dynamic pattern matching implemented in your application, your recipients determine 1) the level of statement detail to view and 2) only the sections of detail that they want to view. Statement data that is not chosen for view by the recipient is simply not extracted, thus presenting statement data faster to the recipient and freeing up system resource in your application environment.

Setting Dynamic Pattern Markers in DefTool

Example 1:

A telecom company wishes to present basic account information and summary data to recipients as the first view of the statement.

Recipients can have multiple Local and/or Long Distance Phone Lines. On the first view of the statement, the Local Line Summary and Long Distance Line Summary sections are shown, with each Phone Line and its charge in the appropriate list. Since dynamic pattern matching was implemented here, a link has been placed around each phone line that will make a second request for live retrieval, this time for the charge detail related to the phone line.

When the recipient clicks on a phone line for which they want to see the charge detail, (such as local line detail charges, as shown below), the local detail charges data for this phone line ONLY is extracted and presented in the recipient's browser.

Example 2:

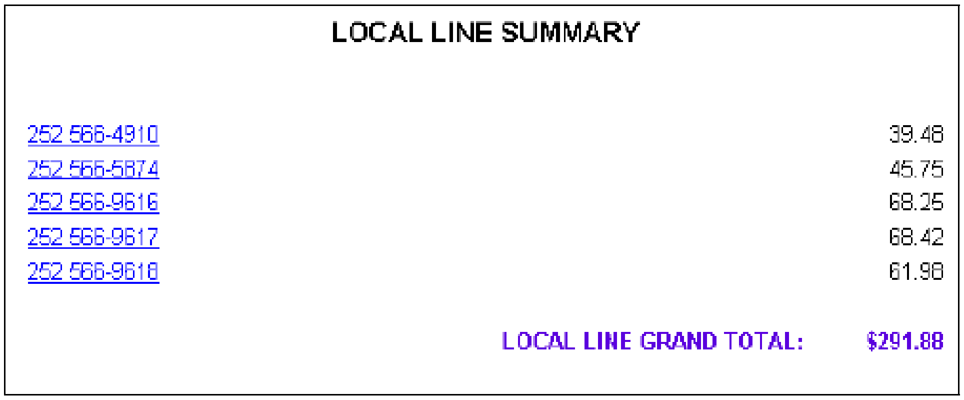
The same telecom company wishes to allow recipients to view portions of their Long Distance Itemized Calls list, based on date, for a selected phone line.

To create the settings for the above scenario in the DefTool, you will create the extraction rules for the detail data (or target) to use a dynamically passed value (the Local Line Phone Number) to extract a specific occurrence of the detail data (detail information pertaining only to the phone line that is selected).

To create settings in DefTool:

1. You must first find the hierarchy in your statement data, and how the different levels of hierarchy relate to one another. For example, the Local Line Summary section of a statement contains basic charge information about a number of local phone \lLines; perhaps it contains the phone line number and current charge for each phone line. The detail data (such as the features, local usage and itemized calls) for each phone line follows later in the statement. Therefore, each item in the summary data is a "parent" to a section of detail data that follows in the statement.
2. Find a data string that is common between:
   - An item in the high-level data that you plan to link FROM (i.e. Summary data), and
   - The item at the marker level (for a group) or table column level (for a table) of the lower-level data section that you plan to link TO (for example, Detail data).

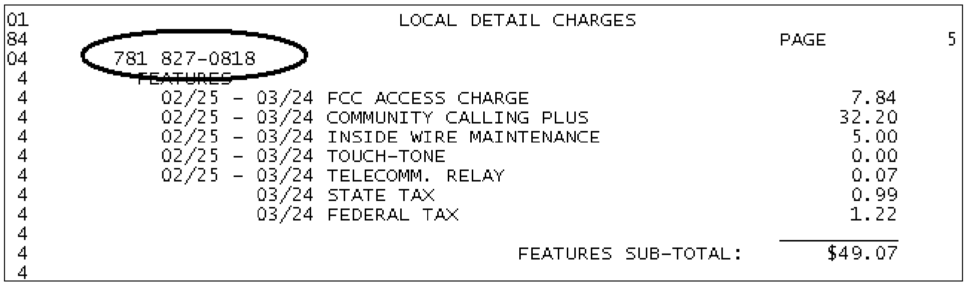
   For example, each item in the Local Line Summary section (shown below) contains a reference to a Phone Line.



LOCAL LINE SUMMARY

252 566-4910	39.48
252 566-5874	45.75
252 566-9616	68.25
252 566-9617	68.42
252 566-9618	61.98
LOCAL LINE GRAND TOTAL:	$291.88

   The Local Charge Detail section for each Phone Line starts with the Phone Line instance (shown below).



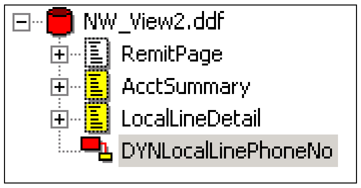
```
01                                  LOCAL  DETAIL  CHARGES
84                                                                        PAGE           5
04     781  827-0818
 4         FEATURES
 4           02/25  -  03/24  FCC  ACCESS  CHARGE                                    7.84
 4           02/25  -  03/24  COMMUNITY  CALLING  PLUS                              32.20
 4           02/25  -  03/24  INSIDE  WIRE  MAINTENANCE                              5.00
 4           02/25  -  03/24  TOUCH-TONE                                             0.00
 4           02/25  -  03/24  TELECOMM.  RELAY                                       0.07
 4                    03/24  STATE  TAX                                              0.99
 4                    03/24  FEDERAL  TAX                                            1.22
 4
 4                                           FEATURES  SUB-TOTAL:                  $49.07
 4
```

CAUTION: If your Target data is an occurrence of a group, the common data string you choose must exist at one of the markers of the section of Detail data you are trying to link to. If your target data is a portion of a table (based on a column constraint, such as all Calls made in the evening), then your common data string must exist in a column of the table.

3 Create your extraction rules for the Detail section (also known as the Target) in the DDF, using the DefTool. Most likely, this type of data will be best extracted as a group with several tables. In the example shown above, a group would be created to extract all Detail data for a Phone Line, starting at the row containing the Phone Line instance itself and ending at the row containing the Phone Line Subtotal data.

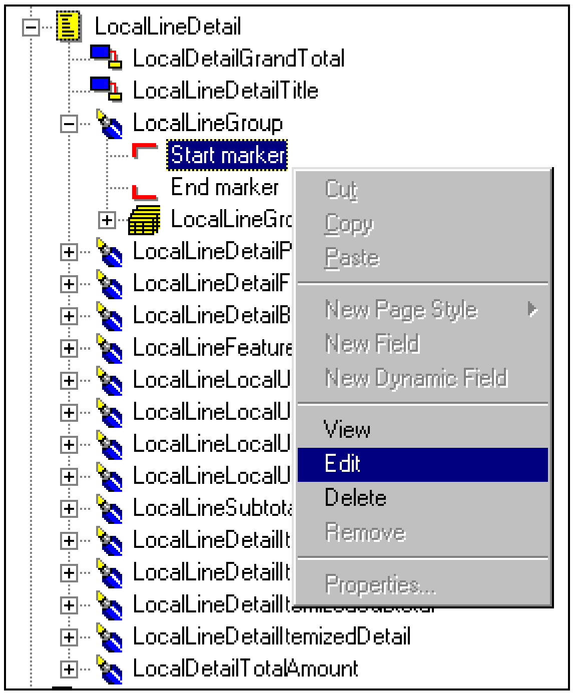
Therefore, the Phone Line example (variable regular expression suitable for validating a phone number string) in the detail group would be the best start marker, and the Subtotal string (i.e. SUB-TOTAL FOR) would be the best end marker for the detail group.

4 Create a new dynamic field in your DDF, by clicking the New Dynamic Field icon in the DefTool toolbar and locating the resulting Dynamic Field node in the tree. It appears in the tree in rename-mode; give it a name that you will recognize later (Such as *DYNLocalLinePhoneNo*.)



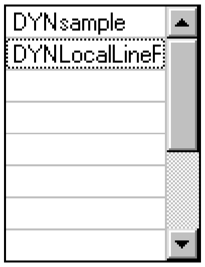
```
NW_View2.ddf
  RemitPage
  AcctSummary
  LocalLineDetail
  DYNLocalLinePhoneNo
```

5 Set the pattern for the marker of your detail group to use a dynamic pattern rather than just a specific regular expression.

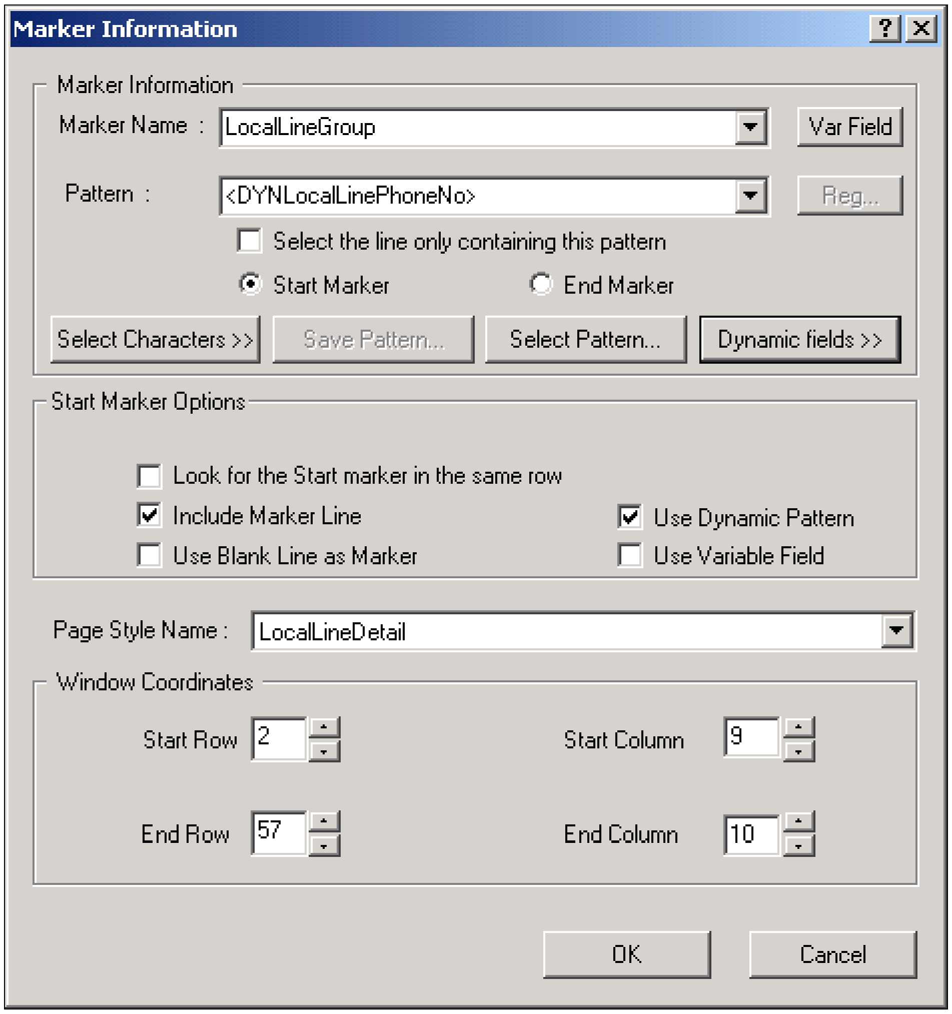


6 In the Marker Properties dialog, Enable the **`Use Dynamic Pattern`** checkbox. (Note that the Test button is now changed to Dynamic Fields.)

7 Click **`Dynamic Fields`** and select your new dynamic field by locating it in the resulting drop down list and double-clicking on it. Your Pattern field should now be populated with the name of your dynamic field enclosed in angle brackets (such as **`<DYNLocalLinePhoneNo>`**).



8 Erase any other non-static characters from the Pattern field, and click **OK** to save changes and close the Start Marker Properties dialog.



Marker Information

Marker Name : LocalLineGroup

Pattern : <DYNLocalLinePhoneNo>

[ ] Select the line only containing this pattern

[✓ Start Marker] End Marker

Start Marker Options

[ ] Look for the Start marker in the same row

[✓] Include Marker Line

[ ] Use Blank Line as Marker

[✓] Use Dynamic Pattern

[ ] Use Variable Field

Page Style Name : LocalLineDetail

Window Coordinates

Start Row 2

Start Column 9

End Row 57

End Column 10

TIP: It may be prudent to leave a partial regular expression in the Pattern field along with the Dynamic Field reference. For example, if the phone line in the detail group was preceded by the string **`LOCAL LINE:`**, then you would set the start marker pattern to locate: **`LOCAL LINE: <DYNLocalLinePhoneNo>`**

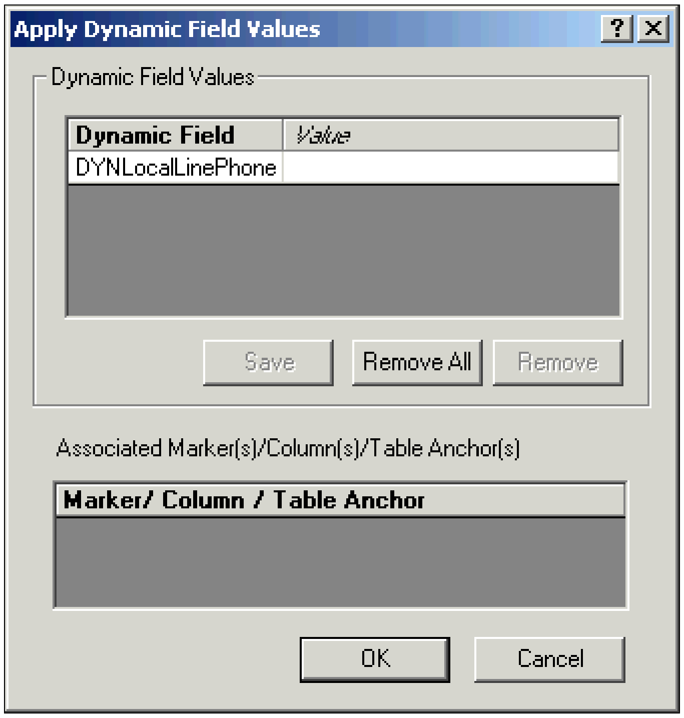
You need to create a hyperlink around the common data item in the high-level view (i.e. the summary view). (In this instance, you can create a link around each Local Phone Line in the LocalLineSummary.) This link, when clicked, will make a specific request for data retrieval and presentment. (In our example, it will request extraction of the LocalChargeDetail group, but only the occurrence of the group where the Phone Line matches the Summary Phone Line that is selected).

Simulating Dynamic Field Values

Continuing with the procedures from above, you can check your work by simulating using the newly created dynamic pattern.

To simulate using dynamic field values:

1 Click . The Apply Dynamic Values dialog appears.


Apply Dynamic Field Values
Dynamic Field Values
Save
Remove All
Remove
Associated Marker(s)/Column(s)/Table Anchor(s)
OK
Cancel

Dynamic Field	Value
DYNLocalLinePhone	

Marker/ Column / Table Anchor

2 Click to select the dynamic field for simulation.

3 Double-click in the **`Value`** field to enter the dynamic value. (Enter 202.)

4 Click **`OK`**. Only extractions for the number “202” will appear.

5 Save the DDF by clicking the Save icon .

Completing the Dynamic Pattern Link in Composer

Now you need to complete the linking in Composer.

To link the dynamic pattern in Composer:

1 Select **`Start>Programs>eStatement>Composer`**.

TIP: All DefTool configurations for dynamic pattern matching are performed at the Target view level in the DDF. All Composer configurations for dynamic pattern matching are performed within the source view’s ALF (i.e., the summary view)

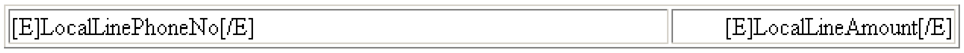
2 Select **`File>Open`**.

3 In the resulting Open dialog, browse to *<EDX_HOME>\Samples\NatlWireless\NW_LocSummary.alf*.

4 Go to the Default Template in the WYSIWYG area of the Composer GUI, and locate the table that contains the column whose data will be wrapped with a hyperlink to selectively extract detail information. (Locate the LocalLineSummary table tag: **`[E]LocalLineSummary,R[/E]`**.)

5 Right-click on the table tag and select **`Configure Table`**. (For National Wireless, right-click **`[E]LocalLineSummary,R[/E]`** and select **`Configure Table`**. The WYSIWYG area displays the ALF's placeholder for the HTML table, **`[E]LocalLineSummary,R[/E]`**.)

6 You can edit the LocalLineSummary table to include static text or image items for linking. For example, the original table configuration in our National Wireless example appears as follows:



[E]LocalLinePhoneNo[/E]	[E]LocalLineAmount[/E]

You can add your own text link such as, "Click here for details," or add a graphic.

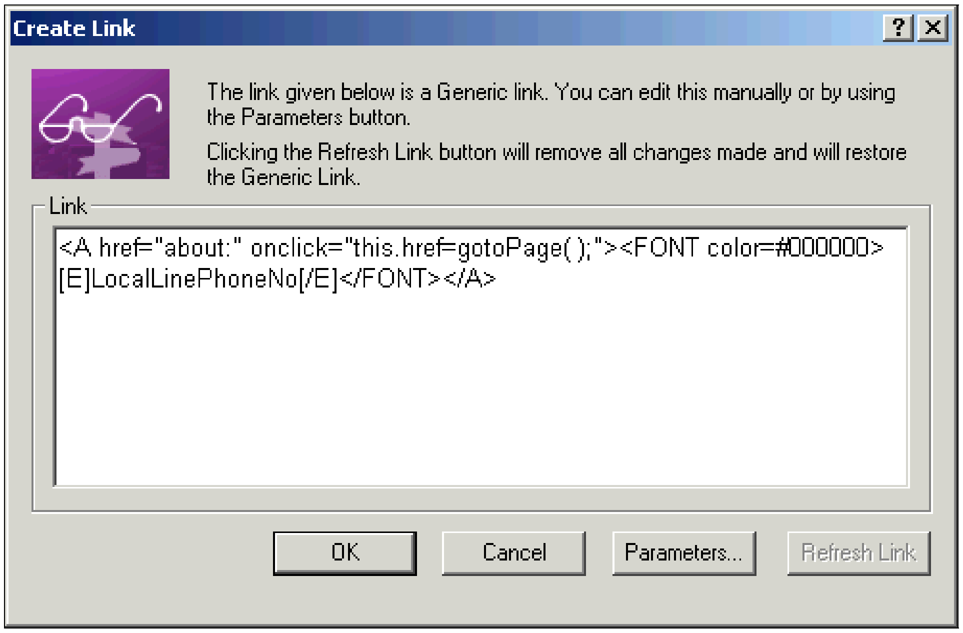
7 Locate the column tag for linking and completely highlight the tag. (Highlight `[E]LocalLinePhoneNo[/E]`.)

CAUTION: It is essential to highlight the entire tag before proceeding, so that your new link is wrapped around the tag properly.

8 Right-click the highlighted tag and select **`Create Link`** from the right-click menu. The Create Link dialog appears, containing generic link syntax.

9 Edit the link to provide the following information:

- The target view (DDF/ALF pair) that contains extraction and presentment rules for the Detail data
- The name of your dynamic field
- The name of the data element that will be replaced by the value inserted into the dynamic field.



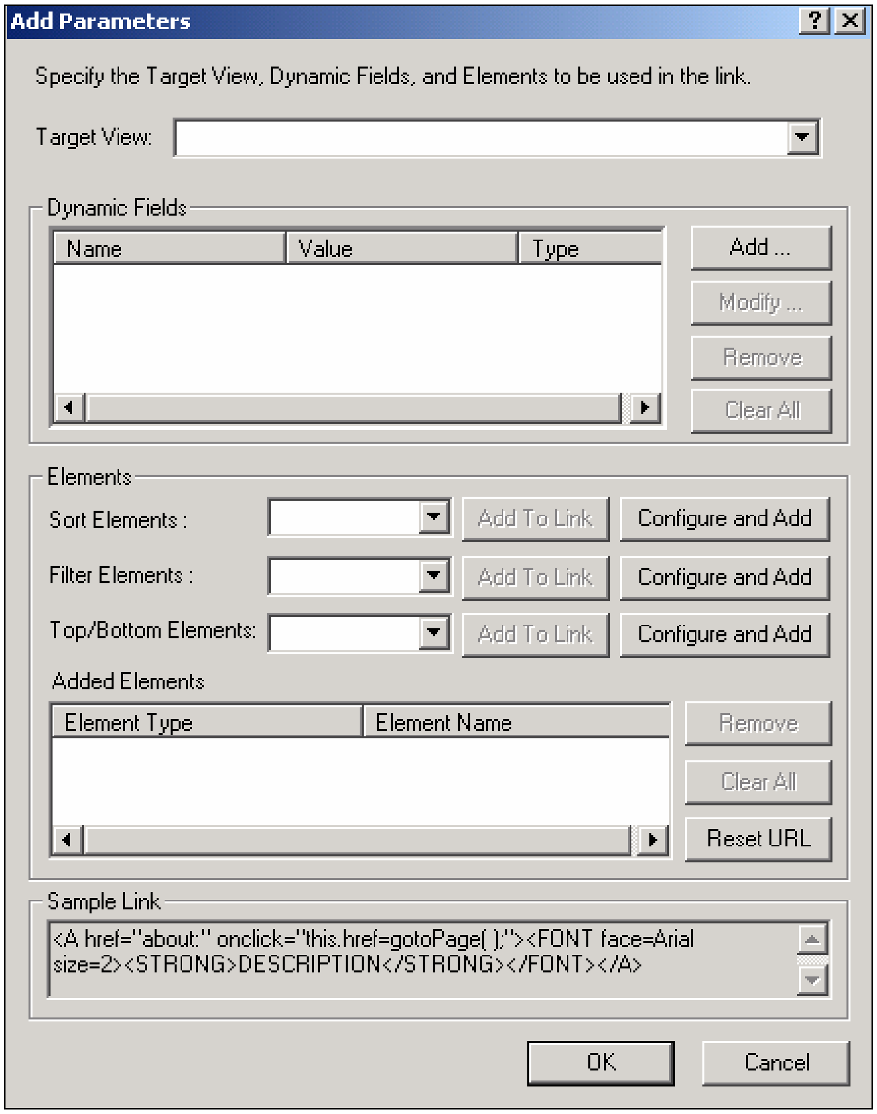
The link given below is a Generic link. You can edit this manually or by using the Parameters button.

Clicking the Refresh Link button will remove all changes made and will restore the Generic Link.

```
<A href="about:" onclick="this.href=gotoPage( );"><FONT color=#000000>
[E]LocalLinePhoneNo[/E]</FONT></A>
```

CAUTION: If the Link to View dialog already has a link in it, click **`Refresh Link`** to remove the link and restore the generic link. You can edit this link directly, or from the Parameters dialog.

10 Click the Parameters button. The Add Parameters dialog appears.



```
<A href="about:" onclick="this.href=gotoPage( );"><FONT face=Arial
size=2><STRONG>DESCRIPTION</STRONG></FONT></A>
```

On the Add Parameters dialog, you must specify:

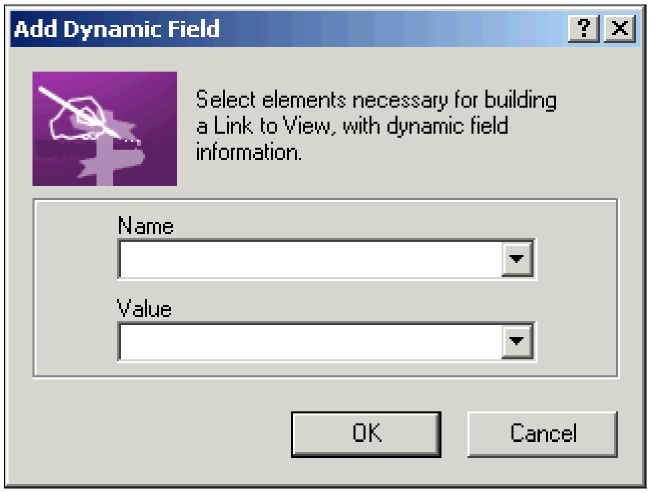
- Your **`Target View`** name
- The **`Dynamic Field`** that is used in the target group marker; when the link is clicked, this “parameter” will be populated with the actual link string and sent back during the request for detail data, to extract only the group occurrence that matches this string.
- Sort, filter, and top/bottom Elements

11 In the Target View field, enter the name of the view that contains extraction and presentment rules for the target, or detail, data (e.g. **`NatlWireless`**).

In the Dynamic Fields section, you will add the parameters that will be used to verify the correct occurrence of detail for presentment. You can specify one parameter, or multiple parameters to certify the correct occurrence of detail data to be presented

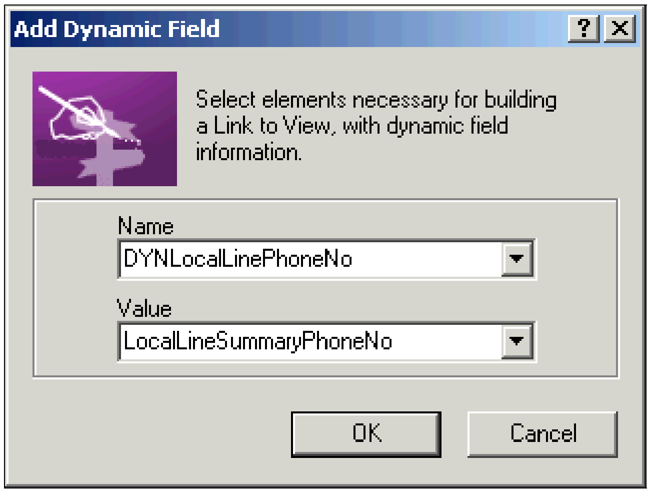
12 Click Add... . The Add Dynamic Fields dialog appears.

Dynamic fields define the requirements for extracting Detail data. The end is your dynamic field end, and the Value is the data that will populate the dynamic field when the link is clicked. The Value can be static or can be a field or column occurrence. Most likely, the Value will be the name of the column you are wrapping the link around, so that the link string is used to match to and extract the correct portion of Detail data.


Add Dynamic Field
Select elements necessary for building a Link to View, with dynamic field information.
Name
Value
OK
Cancel

13 In the Name area, enter the dynamic field name in HTML syntax. (Enter **`DYNLocalLinePhoneNo.`**)

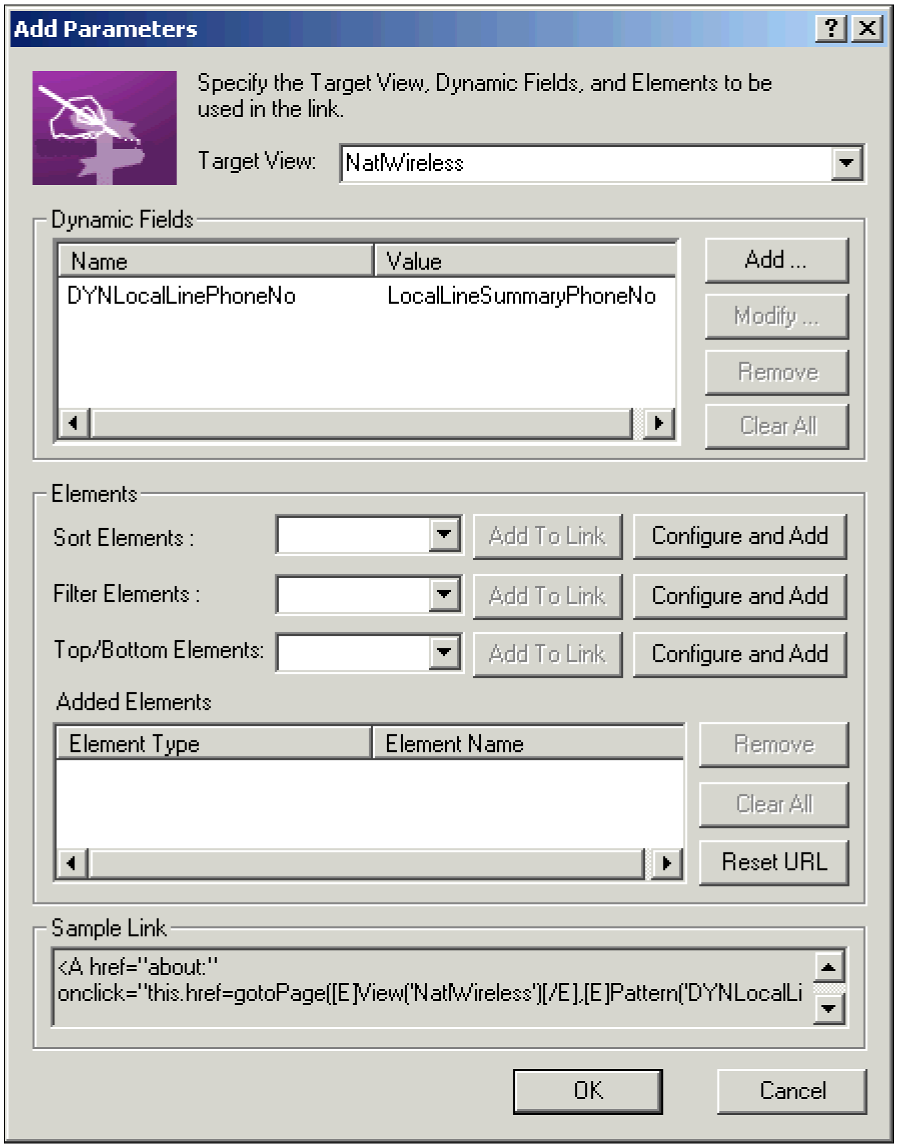
14 In the **`Value`** area, enter the column name that represents data to match the selective extraction in HTML syntax. (Enter **`[E]LocalLineSummaryPhoneNo[/E].`**)


Add Dynamic Field
Select elements necessary for building a Link to View, with dynamic field information.
OK
Cancel

Name: DYNLocalLinePhoneNo

Value: LocalLineSummaryPhoneNo

15 Click **`OK`**. The new name/value pair appears in the Dynamic Fields section of the Add Parameters dialog.


Add Parameters

Specify the Target View, Dynamic Fields, and Elements to be used in the link.

Target View: NatWireless

Dynamic Fields

Name	Value
DYNLocalLinePhoneNo	LocalLineSummaryPhoneNo

Elements

Sort Elements :

Filter Elements :

Top/Bottom Elements:

Added Elements

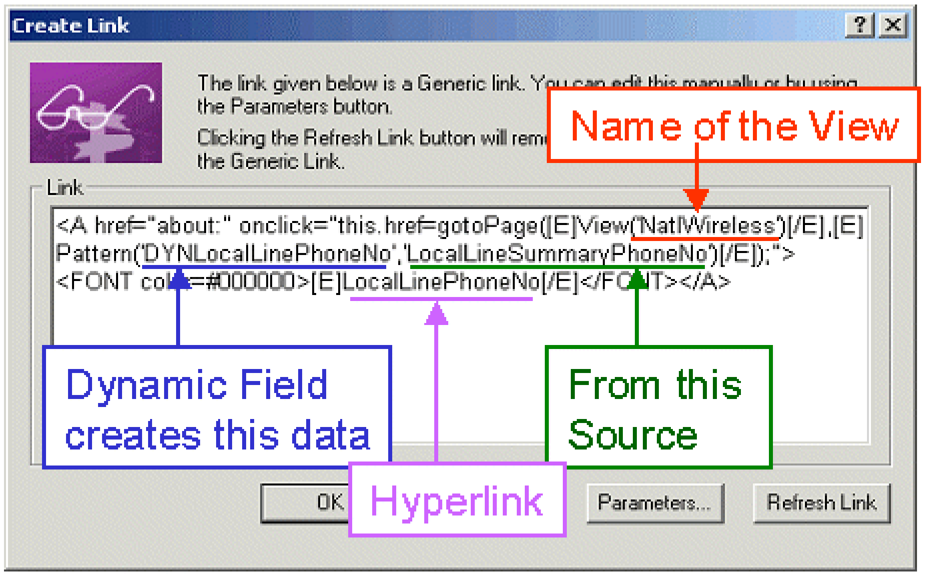
Element Type	Element Name

Sample Link

```
<A href="about:"
onclick="this.href=gotoPage([E]View('NatWireless')[/E],[E]Pattern('DYNLocalLi
```

16 Click **OK** to return to the Create Link window.

View the changes to the hyperlink code that appears in the Link to View dialog. (Below is an example using National Wireless.)

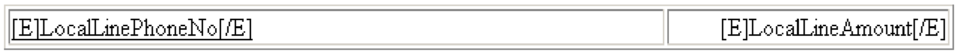

Create Link
The link given below is a Generic link. You can edit this manually or by using the Parameters button.
Clicking the Refresh Link button will remo... the Generic Link.
Link
Name of the View
Dynamic Field creates this data
Hyperlink
From this Source
OK
Parameters...
Refresh Link

```
<A href="about:" onclick="this.href=gotoPage([E]View('NatlWireless')[/E],[E]
Pattern('DYNLocalLinePhoneNo','LocalLineSummaryPhoneNo')[/E]);">
<FONT color=#000000>[E]LocalLinePhoneNo[/E]</FONT></A>
```

This link will not only send a request to see the Detail View of the statement, but only the Detail View that pertains to the Local Line Summary Phone Number that is requested by the link.

TIP: For more information about the JavaScript function that is used in this link, contact the Oracle Training department.

17 Click **OK** to close the Link to View dialog. In the WYSIWYG area of the Composer, you will notice that your table column name is now displayed as a link. This link code, along with other table formatting information, is stored in the ALF and referenced during live retrieval of the Summary data. (An example using National Wireless is shown.)



[E]LocalLinePhoneNo[/E]	[E]LocalLineAmount[/E]

14 Element IDs

Element IDs

Element IDs uniquely identify each item of data in a dynamically composed statement, including each field (standalone, non-repeating data), group, table, and row within each table. eStatement Manager does not assign element IDs to table columns or cells.

Element IDs are transparent to the user, and require customization for any type of implementation with eStatement Manager.

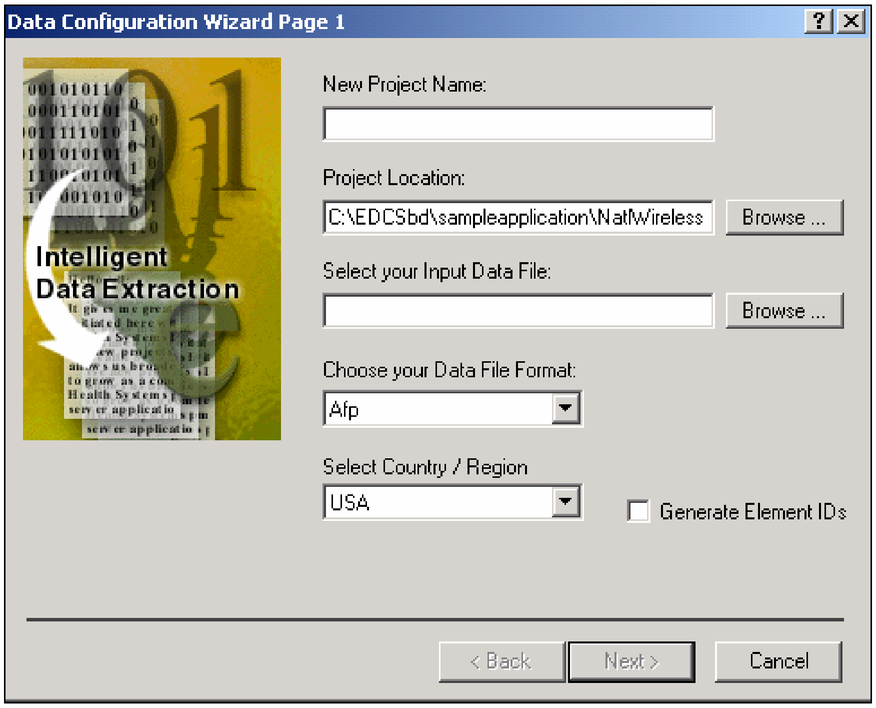
Enable the element ID generation feature in a new or existing DDF if your eStatement Manager application requires unique line item identification in a view, such as for use with a customized annotation and dispute feature or other custom functionality.

If you are not sure whether you need to turn on this option for a new DDF, consult your application designer or leave the feature off. You can enable or disable the feature at any time if necessary.

Anytime you turn the element ID feature on or off in an existing DDF, you must republish the view. See the *Administration Guide for Oracle Siebel eStatement Manager* for details about publishing version sets.

To enable element IDs when creating a new DDF:

- On Page 1 of the Data Configuration Wizard, select the **`Generate element IDs`** option.



New Project Name:

Project Location: C:\EDCSbd\sampleapplication\NatWireless

Select your Input Data File:

Choose your Data File Format: Afp

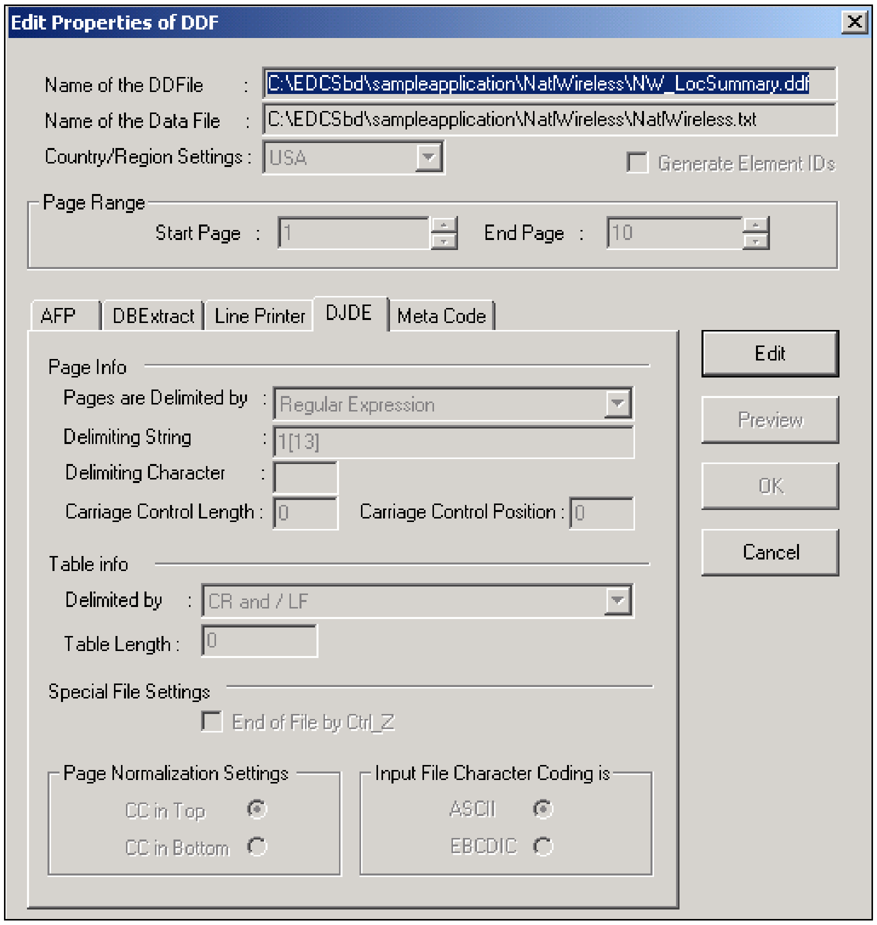
Select Country / Region: USA

☐ Generate Element IDs

To enable/disable element IDs in an existing DDF:

1 Open the statement DDF in DefTool.

2 Select **`Edit>DDF Properties`**. DefTool displays the Edit Properties of DDF dialog.



Edit Properties of DDF

Name of the DDFile : C:\EDCSbd\sampleapplication\NatWireless\NW_LocSummary.ddf

Name of the Data File : C:\EDCSbd\sampleapplication\NatWireless\NatWireless.txt

Country/Region Settings : USA

Generate Element IDs [ ]

Page Range

Start Page : 1 End Page : 10

AFP | DBExtract | Line Printer | DJDE | Meta Code

Page Info

Pages are Delimited by : Regular Expression

Delimiting String : 1[13]

Delimiting Character :

Carriage Control Length : 0 Carriage Control Position : 0

Table info

Delimited by : CR and / LF

Table Length : 0

Special File Settings

End of File by Ctrl_Z [ ]

Page Normalization Settings: [✓ CC in Top] CC in Bottom

Input File Character Coding is: [✓ ASCII] EBCDIC

3 Click the **Edit** button.

4 Enable the **Generate Element IDs** option.

5 Click **OK**.

6 Republish the application view. (See the *Administration Guide for Oracle Siebel eStatement Manager* for details about publishing version sets.)

15 Appendix A: Data Definitions

EBNF Description

EBNF (Extended Backus-Naur Form) is a language described by sets of rules, syntax, tokens etc. Specify one or more syntactic groupings and give rules for constructing the language. For example, in C language, one kind of grouping is called an ‘expression.’ One rule for making an expression might be, “An expression can be made of a minus sign and another expression”.

```
Expression ::= - Expression
```

Rules are often recursive, but there must be at least one rule that leads out of the recursion. EBNF’s repetition is constructed so that an expression has only one derivation.

The most common formal system for presenting such rules that people can read is called EBNF — a context-free grammar.

Data Definitions

Data definitions supported by eStatement Manager for use with the EBNF expression language are:

- **String** — Delimit by double quotes.
  Example: “foo” , “00012”, “508-467-9887”
- **Integer** — Format (+ | - |)(0-9)(0-9)*
  Grouping separator for thousands can be a DCHAR (. or ,)
  Example: 10, 35, 10034, -3462 etc.
- **Double/Float** — Format of (+ | - |)(0-9)(0-9)*. (0-9)(0-9)*
  This is a double precision value, i.e. a decimal value.
  Example: 0.0, -10.23, 10089.4658
- **Date** — Format where DD is the day, MM is the month, YY/YYYY is the year of format (0-9) and Mmm/ Mmmmm is the three-character month name and the full name of the month respectively.
- **Time —** HH stand for hours, MM for minute, and SS for seconds of format (0-9) [AM|PM].
- **Currency —** According to the following format:
  (+ | - | (|) (0-9)(0-9)*. (0-9)(0-9)* (- |) | CR | | DB |)
  In addition, the currency symbol can appear in the data.
  The currency format can change according to the location.

INT(*data*) — Convert the parameter passed as a string to an integer. The characters in the string must all be digits and spaces.

DOUBLE(*data, “decimal separator sign”*) — Convert the parameter passed as a string to double/float value. The characters in the string must all be digits, spaces plus the decimal separator sign.

CURRENCY(*data, "decimal separator sign", "negative format string"*) — Convert the parameter passed as a string to double/float value. The *negative format string* can be "CR", "DB" etc., and the *decimal separator sign* can be ".", "," etc. Data could also have other characters in the string, such as: "$ 1, 000, 000 . 00 CR".

Currency Example	**Notes**
4,123,978.999-	Trailing negative + three decimals
4,123,978.99-	Trailing negative
-4,123,978.99	Leading negative
-4,123,978.999	Leading negative + three decimals
4,123,978.99CR	Trailing CR (credit)
4,978.99 CR	Trailing CR separated by space
4,123,978.999CR	Three decimals, trailing CR (credit)
4,978.999 CR	Three decimals, trailing CR separated by space
4,123,978.99DB	Trailing DB (Debit)
4,978.99 DB	Trailing DB separated by space
4,123,978.999DB	Three decimals, trailing DB (Debit)
4,978.999 DB	Three decimals, trailing DB separated by space

DATE(*data, format_string*) — Convert the parameter passed as a string to Date value.

TIME(*data, format_string*) — Converts the parameter passed as a string to the Time value.

Date/Time Format String Components			
Format	**Description**	**English**	**French**
%%	Parse literal '%'		
%a	Day of week using the locale's weekday names (abbreviated name)	Fri	ven
%A	Day of week using the locale's weekday names (long name)	Friday	vendredi
%b	Month using the locale's month names (abbreviated name)	Jun	jun
%B	Month using the locale's month names (full name)	June	juin
%c	Locale-appropriate date and time representation (abbreviated form)	Fri Jun 08 17:53:28 2002	ven 08 jun 2002 17:50:58 EDT
%C	Locale-appropriate date and time representation (full form)		
%d	Day of month [1-31]; leading zero is permitted but not required	8 08	
%D	Date as %m/%d/%y.		
%e	Same as %d.		
%G	One digit of fractional seconds. Leading zero is permitted but not required. Fractional second value increments in six second intervals; e.g., .1=6 seconds, .7=42 seconds	.6 or 0.6 both equivalent to 0:36	
%h	Same as %b.	Jun	jun
%H	Hour (24-hour clock) [0-23]. Leading zero is permitted but not required.		
%I	Hour (12-hour clock) [1-12]. Leading zero is permitted but not required.		
%j	Day number of the year [1-366]. Leading zeros are permitted but not required.		

Date/Time Format String Components			
Format	**Description**	**English**	**French**
%K	Time duration as HHHH. Leading zeros are permitted but not required.	1324 0123	
%m	Month number [1-12]. Leading zero is permitted but not required.		
%M	Minute [0-59]. Leading zero is permitted but not required.		
%p	Locale's equivalent of either a.m. or p.m.	For U.S. locale: AM, PM, am, pm	
%Q	Time duration as MMM. Leading zeros are permitted but not required.	324 12 012	
%r	Appropriate time representation in the 12-hour clock format with %p.		
%R	Time as %H:%M.		
%S	Seconds [0-61]. Leading zero is permitted but not required. The range of values is [00-61] rather than [00-59] to allow for the occasional leap second and even more infrequent double leap second.		
%T	Time as %H:%M:%S.		
%w	Weekday as a decimal number [0-6], with 0 representing Sunday.		
%x	Locale-appropriate date representation.	06/08/02	08.06.2002
%X	Locale-appropriate time representation.	17:50:03	17:50:03

Date/Time Format String Components			
Format	**Description**	**English**	**French**
%y	The year within a century [00-99]. When a century is not otherwise specified, values in the range 69-99 refer to years in the twentieth century (1969 to 1999, inclusive). Values in the range 00-68 refer to years in the twenty-first century (2000 to 2068 inclusive). Leading zeros are permitted but not required.	02	02
%Y	Year including the century (for example 1993) [0001-9999].	2002	2002

Date String Formats		
Input Format	**Examples**	**Format String**
DDMMYYYY	31012002	`%d%m%Y`
YYYYMMDD	20021130	`%Y%m%d`
YYYY	2002	`%Y`
MMDD	1130	`%m%d`

YYYY-MM	2002-06
	2002/06
	2002.06
		`%Y-%m`
		`%Y/%m`
		`%Y.%m`

YYYY-MM-DD	2002-06-30
	2002/06/30
	2002.06.30
		`%Y-%m-%d`
		`%Y/%m/%d`
		`%Y.%m.%d`

DD.MM.YY	30.06.02	`%d.%m.%y`
DD.MM.YYYY	30.06.2002	`%d.%m.%Y`

DD-Mmm-YYYY	30-Jun-2002
	30.Jun.2002
	30 Jun 2002
		`%d-%b-%Y`
		`%d.%b.%Y`
		`%d %b %Y`

DD-Mmm-YY	30-Jun-02
	30.Jun.02
	30 Jun 02
		`%d-%b-%y`
		`%d.%b.%y`
		`%d %b %y`

DD Mmm Space between day and month	30 Jun	`%d %b`
DD/MM/YY	30/06/02	`%d/%m/%y`
DD/MM/YYYY	30/06/2002	`%d/%m/%Y`
DD Mmmmm YYYY Includes full name of month, space between day and month, month and year	30 April 2002	`%d %B %Y`

Time String Formats		
Input Format	**Examples**	**Format String**
HHMM	2344	`%H%M`
HH:MM	23:44	`%H:%M`
HHMM PM Space before meridian indicator	1144 PM 1144 AM 1144 am	`%H%M %p`
HHMMPM	1144PM 1144AM 1144am	`%H%M%p`
HH:MM PM Space before meridian indicator	11:44 PM 11:44 AM 11:44 am	`%H:%M %p`
HH:MMPM	11:44PM 11:44AM 11:44am	`%H:%M%p`
HHMMSS	234405	`%H%M%S`
HH:MM:SS	03:44:05 22:10:36	`%H:%M:%S`
HHMMSS PM Space before meridian indicator	114405 PM 114405 pm 114405 am	`%H%M%S %p`
HHMMSSPM	114405PM 114405pm 114405am	`%H%M%S%p`
HH:MM:SS PM Space before meridian indicator	03:44:05 PM 03:44:05 pm 03:44:05 am	`%H:%M:%S %p`
HH:MM:SSPM	03:44:05PM 03:44:05pm 03:44:05am	`%H:%M:%S%p`

Time Duration String Formats		
Input Format	**Examples**	**Format String**
HH:MM**.**D Duration of call in hours, minutes, and decimal minutes (0.1 = 6 seconds)	13:06.1	**`%H:%M.%G`**
MMM:SS	135:05	**`%Q:%S`**
HHHH:MM:SS	1233:05:17	**`%K:%M:%S`**

Index

A

AFP font settings, 46
ALF. *See Application Logic File*
Application Logic File
  definition, 14
Application, sample, 15

C

Chart
  extracting tables, 97
Customer Self-Service, 9, 133

D

Data
  definitions, 169
Data Definition File, 13
  for DJDE formats, 35
  for line printer formats, 50
Data extraction, 12, 17
  area, 57
  creating rules, 62
  data pattern, 59
DDF. *See Data Definition Files*
Definition toolbar, 27
DefTool, 15
  icons, 27
  menu, 20
  simulations, 24
  toolbars, 27
Document
  define style, 65
  recipient, 68
  start, 63
Document style, 27
Dynamic field, 28
Dynamic pattern matching, 153
  link, 158
  set markers, 154
  simulating values, 158

E

eaSuite, 9, 133
EBNF Description, 169
Electronic statement presentment, 11
Element IDs, 165
eStatement
  application files, 15
  application process, 13
  development process, 13
eStatement Manager
  introduction, 11
Extraction simulator, 133

F

Field
  anchor, 89
  create anchor, 92
  define, 74, 78, 81, 82, 83
  delete anchor, 95
  edit anchor, 95
  extraction simulation, 135
  properties, 60

G

Group
  components, 112
  define, 111, 114
  define end marker, 119
  define properties, 122

define start marker, 116
new, 30
table columns, 113
Group fields, 131

H

Help, 10
Highlighting data, 19
History list
sub-account data, 125

I

Indexing sub-account data, 125

M

Marker
new, 29
properties, 61
Metacode
settings, 54

N

National Wireless sample application, 125
National Wireless Sample Application, 15
New field, 28

P

Page style, 28, 72
define, 72
Presentation design, 12
Presenting sub-account data, 125
Primary key
define, 69

R

Regular expression, 141
currency values, 145
longhand dates, 146
partially static data, 149
shorthand dates, 145
static data, 148
varying data values, 147

S

Sample application
National Wireless, 15
sub-account view, 125
Selective extraction
simulation, 137
Simulating
dynamic field values, 25
Simulator
navigation, 133
Statement
mastering, 11
Sub-account
creating a DDF, 125
inclusion in history list, 125
Subkey, 131

T

Table
anchor, 61
chart extraction, 97
column, 62
define anchor, 109
define column, 107
define end marker, 103
define external anchor, 109
define format, 106
define internal anchor, 110
define start marker, 100
definitions, 98
external anchor, 62
extracting, 97
extraction simulation, 136
internal anchor, 62
new, 29
properties, 61
rules, 99
Template, 14

V

View, 12
  sub-account, 125

Z

Zoom in/out, 19