SOLUTIONS

Microsoft Dynamics NAV

What’s New:
Developing Solutions for Microsoft Dynamics™ NAV “6.0”

White Paper

November 2007
www.microsoft.com/dynamics/nav

Table of Contents

Introduction

This paper describes the changes in Microsoft Dynamics™ NAV "6.0" and is intended for Microsoft Dynamics NAV developers and salespeople who have used previous versions of Microsoft Dynamics NAV.

Microsoft Dynamics NAV "6.0" leverages your existing investment in the Microsoft Dynamics NAV product and extends how customers can do business in the following ways:

- **Customizable, Role-Based User Interface** — Microsoft Dynamics NAV "6.0" provides a more intuitive and customizable user interface (UI) called the RoleTailored client, which you can modify to support the job functions of different work roles in your organization. For each role that you support, you can create a customizable Role Center that displays key information that is required by that role and makes their day-to-day tasks easier to complete. For example, you can customize the sales order processor Role Center to display an interactive list of customers, daily activities, and sales information and also give users easy access to their Microsoft Office Outlook® e-mail, tasks, and calendar. Partners, administrators, and super users can customize the UI for specific roles, and individual users can also further personalize the UI to meet their specific needs.
- **Same Data, Different Clients** — Microsoft Dynamics NAV "6.0" includes a new object called a *page*. Pages are used in the same way as forms. However, unlike forms, pages can be displayed on multiple clients, including the RoleTailored client, a Web browser that displays Microsoft SharePoint® Services, or other custom clients. This lets you display the same information from a Web site as you can display from a desktop client without writing additional code.
- **Improved Report Design and Functionality** — Microsoft Dynamics NAV "6.0" offers an improved reporting experience. You can now create interactive reports to which you can add charts, graphs, tables, and other data presentation elements that were not available in previous versions of Microsoft Dynamics NAV. You can also save your reports in .pdf or .xls file formats.
- **Business Connections with Web Services** — Web services are a standard, widely used method for integrating applications and are supported in Microsoft Dynamics NAV "6.0". By implementing Web services, you can access Microsoft Dynamics NAV data and business logic from outside the product in a standard, secure way. This enables you to connect Microsoft Dynamics NAV to other systems within an organization.

In this white paper, the following major areas are described:

- Architectural changes from a two- to a three-tier architecture, including a discussion of the new RoleTailored client and Microsoft Dynamics NAV Server with Web services support.
- Development improvements and benefits for Microsoft Dynamics NAV "6.0", including the introduction of the new Page Designer, changes in Object Designer, and enhanced reporting using SQL Server® Reporting Services.
- Functionality that needs to be redesigned from a previous version to work in Microsoft Dynamics NAV "6.0".
- Step-by-step walkthroughs that demonstrate new development features, including creating and modifying pages and creating a report.
- Appendix with a list of elements that have changed in the three-tier architecture.

Architectural Changes in Microsoft Dynamics NAV "6.0"

In previous versions of Microsoft Dynamics NAV, the architecture consisted of two tiers: a client and a server. The client was responsible for both displaying the presentation layer in forms and running the business logic in C/AL code.

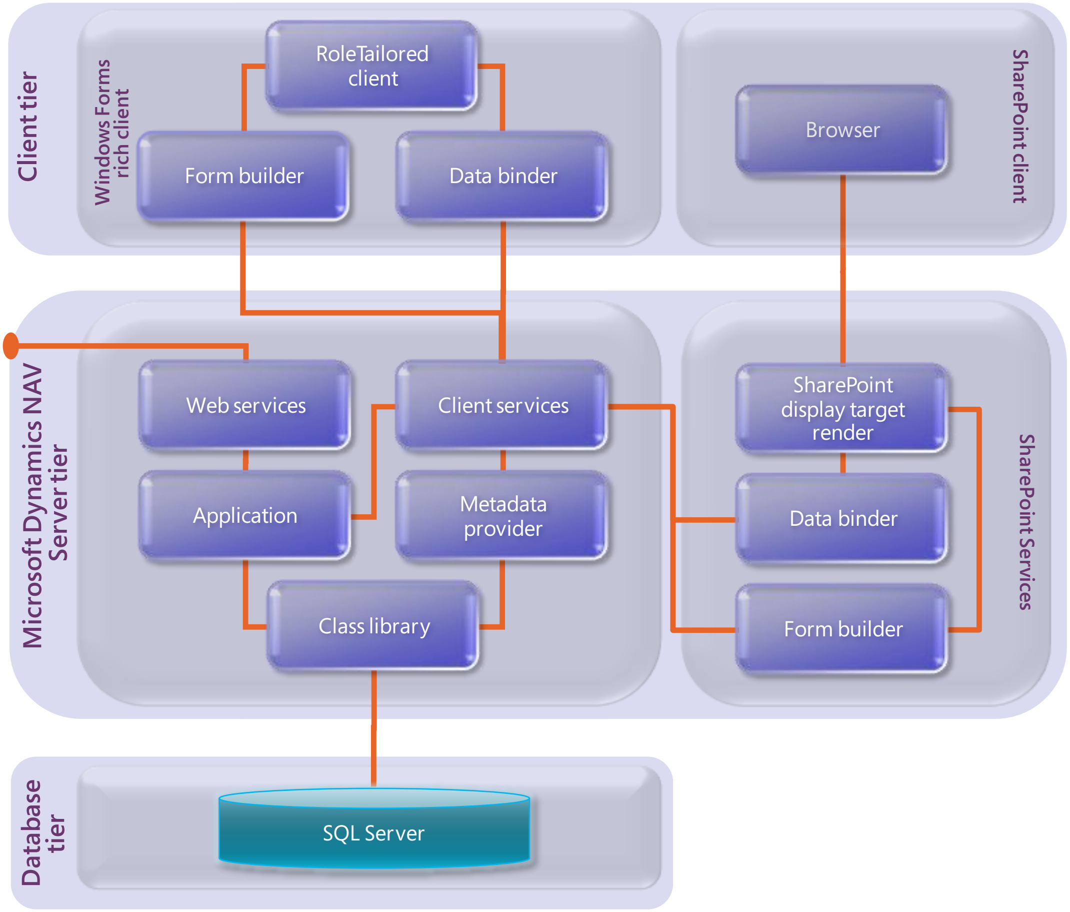
The architecture of Microsoft Dynamics NAV "6.0" includes a two-tier model, where the Classic client provides a front end to the data layer, which is either the Database Server for Microsoft Dynamics NAV Classic or SQL Server.

The architecture has also been extended to support a three-tier model, with:

- **RoleTailored client**: A client user interface that provides the front end. Users can also access information through a Web-based client that runs on Microsoft SharePoint Services.
- **Microsoft Dynamics NAV Server**: A middle-tier server that provides the business logic layer.
- **SQL Server**: A database server that provides the data layer.

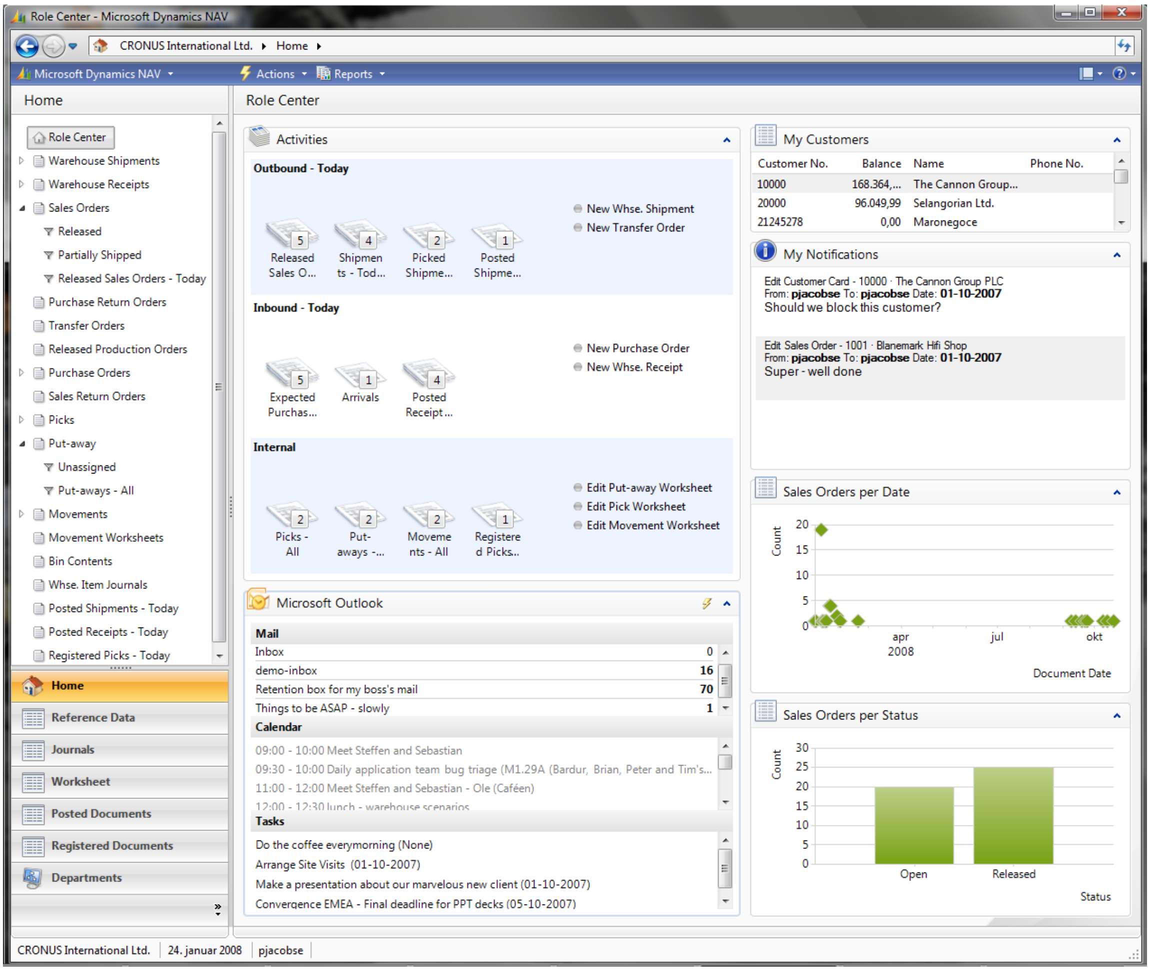
If you are running the Database Server for Microsoft Dynamics NAV Classic, then you must move your system to the SQL Server Option to migrate to the three-tier architecture, with the RoleTailored client and Microsoft Dynamics NAV Server.

The following figure describes the three-tier architecture in Microsoft Dynamics NAV "6.0":



RoleTailored Client

The RoleTailored client, which is a Windows Forms–based client, has been designed as the front end for the new architecture. One key feature of the RoleTailored client is the new user interface. After extensive research into understanding how customers work, what their business goals are, and how their departmental organization works, Microsoft has created a new interface with new interaction styles that support working in a role-oriented, task-focused manner.



Customer No.	Balance	Name	Phone No.
10000	168.364,...	The Cannon Group...	
20000	96.049,99	Selangorian Ltd.	
21245278	0,00	Maronegoce	

Edit Customer Card - 10000 · The Cannon Group PLC
From: **pjacobse** To: **pjacobse** Date: **01-10-2007**
Should we block this customer?

Edit Sales Order - 1001 · Blanemark Hifi Shop
From: **pjacobse** To: **pjacobse** Date: **01-10-2007**
Super - well done

Mail

Inbox	0
demo-inbox	16
Retention box for my boss's mail	70
Things to be ASAP - slowly	1

Calendar

09:00 - 10:00 Meet Steffen and Sebastian
09:30 - 10:00 Daily application team bug triage (M1.29A (Bardur, Brian, Peter and Tim's...
11:00 - 12:00 Meet Steffen and Sebastian - Ole (Caféen)
12:00 - 12:30 lunch - warehouse scenarios

Tasks

Do the coffee everymorning (None)
Arrange Site Visits (01-10-2007)
Make a presentation about our marvelous new client (01-10-2007)
Convergence EMEA - Final deadline for PPT decks (05-10-2007)

To support this user interface, Microsoft Dynamics NAV displays data in the RoleTailored client in a model-driven way and relies on metadata to display this data. Also, business logic code is no longer run on the client. Instead, requests to run C/AL code are sent to the middle tier.

Previous versions of Microsoft Dynamics NAV and the Classic client in Microsoft Dynamics NAV “6.0” display data in forms, such as card forms or list forms. You design forms in a WYSIWYG environment with precise layout information.

The new RoleTailored client now displays data in *pages*, and you design these pages in Page Designer. The page elements are listed in their relative display order, and the properties that are associated with each element are used to specify special presentation features but without any size or position specifications. This allows a page to be consumed by different clients without being constrained by layout limitations, and each client can render the page in a different way.

Pages provide a flexible foundation for building many different types of display objects. The Classic client contains card forms and list forms. In the RoleTailored client, there are equivalent page types and several new ones that help you build pages with special layouts. Pages also contain new types of controls that enable advanced representation of system data and shortcuts to system features. Pages also contain fewer triggers than forms for two reasons:

- The RoleTailored client has enhanced behavior that is implemented directly into the controls that run on it. Some code that was previously necessary is now obsolete because of these new controls.
- Pages have been designed to optimize performance, which reduces requirements for all existing form triggers.

For more information about pages, see Page Designer in the Development Enhancements in Microsoft Dynamics NAV "6.0" section.

The RoleTailored client provides a single document interface (SDI) environment, which is the same kind of environment that is used in Outlook. Lists of e-mails, tasks, or contacts are displayed in the navigation layer, and every e-mail that you want to read or write opens in its own window.

In the RoleTailored client, the navigation layer displays lists of work areas, which are called list places. Examples of list places include customers and orders. By default, one list replaces another list in the same window. You can also launch a list or card page in its own window. In addition to list places, the navigation layer shows the Role Center and Departments place.

You then open an entity, such as a specific customer or sales order, in a separate window to view or edit it. These windows are called task pages, which also fall into subcategories with different layouts, such as cards, documents, and journals.

The RoleTailored client also has a small category of dialog boxes that are different from task pages. These dialog boxes differ in layout and do not represent an entity. Instead, they prompt the user for parameters for further processing.

Microsoft Dynamics NAV Server

The new Microsoft Dynamics NAV Server is a .NET-based Windows Service application that works exclusively on SQL Server. It uses the Windows Communication Framework as the communication protocol for RoleTailored clients and Internet Information Services for Web services and Microsoft SharePoint clients that connect with it. It can execute multiple client requests in parallel and serve other clients by providing Web service access to authenticated clients.

A key difference between the two-tier architecture with the Classic client and three-tier architecture with the RoleTailored client is that business logic runs on Microsoft Dynamics NAV Server instead of the client in the three-tier architecture. A simple example is the FILE.CREATE function. In previous versions of Microsoft Dynamics NAV and in the Classic client, when code such as this code is run, files are created on the client. With the RoleTailored client and Microsoft Dynamics NAV Server, the files are created on the service itself. If the file is meant for a user, then you must take advantage of new C/AL file functions that enables the file to download to the client computer.

Microsoft Dynamics NAV Server provides an additional layer of security between the clients and the database. It leverages the authentication features of the Windows Communications Framework to provide another layer of user authentication and uses impersonation to ensure that business logic is executed in a process that has been instantiated by the user who submitted the request. This means that authorization and logging of user requests is still performed on a per-user basis. This ensures that all Windows authentication and Microsoft Dynamics NAV roles and permissions that have been granted to the user are correct. It also ensures that business logic–level auditing is still performed.

The new architecture provides greater scalability by letting you install multiple machines that are running Microsoft Dynamics NAV Server to provide access the same database and also allow you to balance your system by dedicating different services to particular application areas or to different clients.

Web Services Support

Web services are a standardized way for independent software systems to communicate with each other over standard internet protocols, and the architecture is designed for dynamic program-to-program interaction. In the Web services architecture, many types of distributed systems can be implemented. Examples of distributed systems include synchronous and asynchronous messaging systems, distributed computational clusters, mobile-networked systems, grid systems, and peer-to-peer environments. The broad spectrum of requirements for program-to-program interactions means that the protocols used by Web services are more flexible than Web protocols. However, like the World Wide Web, Web services also rely on a small number of specific protocols, such as Simple Object Access Protocol (SOAP).

Web services are designed specifically to facilitate the highly dynamic data interchange that is required in business transactions. Standardized integration technologies such as Web services bring value to businesses by breaking down data silos that are created by proprietary integration options. These proprietary integration technologies make it difficult to get information in and out of the different systems.

To provide a robust development and operational environment, Web services are described using machine-readable *metadata*. Web service metadata serves several purposes. The metadata is used to describe the message interchange formats that a Web service can support as well as the valid message exchange patterns of a service. Metadata is also used to describe the capabilities and requirements of a service. Web Services Description Language (WSDL), which is an XML-based language for defining Web services, is used to express the interchange formats and message exchange patterns of the Web services

With the introduction of Microsoft Dynamics NAV Server, Microsoft Dynamics NAV supports Web services, which makes it easy to integrate Microsoft Dynamics NAV with other systems. Specifically, Web service integration with Microsoft Dynamics NAV is facilitated through Web service–enabled codeunits and pages. With proper authentication and authorization, external systems can read and write data on pages and call codeunits as defined by the common Web service protocols. The Web service capabilities in Microsoft Dynamics NAV help customers reap the benefits of a service-oriented architecture (SOA).

Microsoft Dynamics NAV Web services are immediately useful to customers and partners who want to use business logic or use a standard interface to access data from outside Microsoft Dynamics NAV. You can use most major software development environments, such as Microsoft Visual Studio® 2005, to build applications that use Web services. Furthermore, because Web services are XML based, you can also build Web services across platforms and programming languages.

Web Services Published by Microsoft Dynamics NAV

There are a few simple types of Web services that can be published by Microsoft Dynamics NAV. The types of Web services correspond to the degree of complexity required in the Web service interface. All Web services can run C/AL code and validation triggers.

- The simplest Web service types are developed using the page object. Microsoft Dynamics NAV constructs a default Web service with a fixed set of methods. In general, it corresponds to general data access, such as get and set individual values or retrieve and update lists of values.

- Another Web service type involves including codeunits and the functions from those codeunits in the Web service.
- The last type of Web service includes the ability to pass complex data types by using an XMLport object as a parameter in a codeunit function.

The way in which Microsoft Dynamics NAV integrates Web services saves you from the complexity of manually setting up Web service frameworks, such as managing the WSDL descriptions. To publish any type of Dynamics NAV Web service (page or codeunit), you add it to the Web Service table.

Pages and codeunits that are added to the Web Service table in Microsoft Dynamics NAV and are published are immediately available for Web service requests over the network. Consumers of these Web services, which are systems integrating with Microsoft Dynamics NAV, only need to know the network name (or address) of the computer that is running Microsoft Dynamics NAV Server and the names given to the individual pages and codeunits. For example, if a computer that is running Microsoft Dynamics NAV Server is named NAV_Server1, the MyCustomer Web service is available at the following URL:

```
http://NAV_Server1/NavisionServer/CRONUS_International_Ltd/Page/MyCustomer.
navws
```

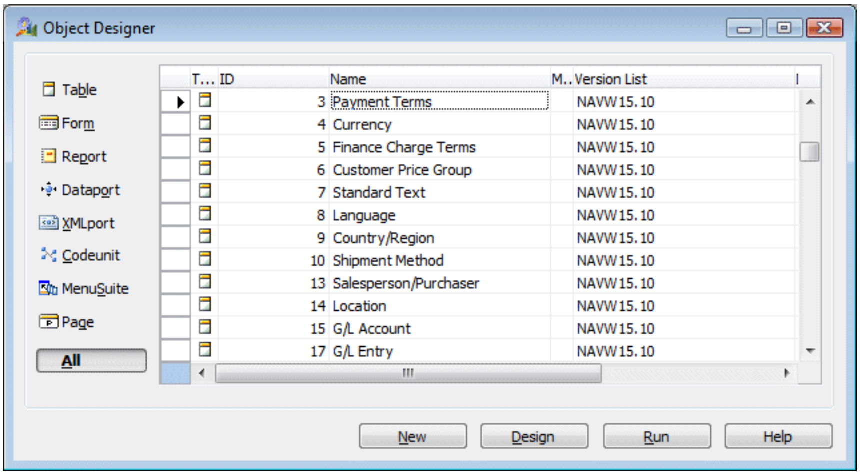
It is important to know that Microsoft Dynamics NAV manages Web service requests in the same way as it handles requests from end users. Access rights authorization and validation, input data validation, business logic invocation, and concurrency control are all managed in the same way as requests from a Microsoft Dynamics NAV client. This ensures that the integrity of the Microsoft Dynamics NAV data is not compromised by using Web services. It also means that you do not have to replicate code that validates data or invokes business logic when you build systems that use the Web services provided by Microsoft Dynamics NAV.

Note that when you are writing code for Web services, you must avoid using any end-user confirmation dialog boxes or message boxes. The code for Web services cannot interact with the client that called the code, so it cannot respond to dialog box or other client interaction requests. Running this code causes an exception to be thrown. The exception can be caught and handled, but the Web service task will not complete.

For more information about developing Web services, see Web Services in the Development Enhancements in Microsoft Dynamics NAV “6.0” section.

Development Enhancements in Microsoft Dynamics NAV "6.0"

In Microsoft Dynamics NAV "6.0", you continue to use Object Designer to develop solutions. You can access new development environment features such as Page Designer from Object Designer in the Classic client.



T...	ID	Name	M..	Version List
	3	Payment Terms		NAVW15.10
	4	Currency		NAVW15.10
	5	Finance Charge Terms		NAVW15.10
	6	Customer Price Group		NAVW15.10
	7	Standard Text		NAVW15.10
	8	Language		NAVW15.10
	9	Country/Region		NAVW15.10
	10	Shipment Method		NAVW15.10
	13	Salesperson/Purchaser		NAVW15.10
	14	Location		NAVW15.10
	15	G/L Account		NAVW15.10
	17	G/L Entry		NAVW15.10

Object Comparison Between Architectures

The following table compares how objects work in the Classic client and RoleTailored client.

Object	Classic client	RoleTailored client
Codeunit	Same as previous versions of Microsoft Dynamics NAV.	Designed and executed in the same way as the Classic client. New functions have been added to stream files and data between the RoleTailored client and Microsoft Dynamics NAV Server. Code runs on Microsoft Dynamics NAV Server. Any applications that need to allow clients access to COM objects need to be redesigned.
Dataport	Same as previous versions of Microsoft Dynamics NAV.	Not supported.
Form	Same as previous versions of Microsoft Dynamics NAV. All code that is written on forms continues to run. Code and triggers run on the client.	Not supported. Replaced by pages. C/AL code references to forms are translated to the same method call on a page object with the same ID.

MenuSuite	Same as previous versions of Microsoft Dynamics NAV.	Navigation based on page rather than MenuSuite. To navigate to another page, you add actions to a page that opens the new page. The Actions menu is similar to using menu buttons to provide links to new forms. You can also allow access to the traditional MenuSuite through the Departments section. Users open a list first instead of a card to more easily find and open a specific entry.
Page	Not supported.	New object type that contains properties and methods to replace forms. Code and triggers run on Microsoft Dynamics NAV Server.
Report	Same as previous versions of Microsoft Dynamics NAV. Reports have three new properties that are ignored.	Report datasets, sections, and request forms or pages are designed in the C/SIDE Report Designer. Report layout is designed with a report definition language editor such as Visual Studio. The layout information is stored in the new properties in the report object.
Table	Same as previous versions of Microsoft Dynamics NAV. A new `ExtendedDataType` property has been added but has no affect on table behavior. Code and triggers run on the client.	Designed and executed in the same way as in the Classic client, with the same triggers and properties. The new `ExtendedDataType` property allows you to add metadata to fields to change how they are rendered. Code and triggers run on Microsoft Dynamics NAV Server.
Web service	Not supported.	New object type that are designed by creating or reusing a page or page and codeunit objects together. XMLports can be used to send complex data to a Web service by making the XMLport a parameter to a codeunit and including that codeunit in a Web service. Published by adding public references to the Web Service table.
XMLport	Same as previous versions of Microsoft Dynamics NAV. The functions for using XMLports as dataports are not available.	Designed and executed in the same way as the Classic client. XMLports have been extended with similar functionality to dataports to allow import and export of structured files.

Page Designer

The RoleTailored client displays data in pages, which you design with Page Designer. Microsoft Dynamics NAV “6.0” comes with the following common types:

Pages in the Navigation Layer

- Role Center page — This page type is used to build the starting page that users see when they start the application. There is a different Role Center page definition for each user profile in the application, although multiple users can be assigned to the same user profile). This page type consists of a number of FactBoxes, which are smaller windows with information related to the current main window. Users can hide or show individual FactBoxes on their Role Center page.
- List page — This page type is the equivalent to the list form in the Classic client. A list page also includes an Action Pane at the top for promoted frequent commands, a filter pane in the top section of the page, and a FactBox pane to the right.
- Departments page — This is a system-generated page type. You define the content of the application with the MenuSuite, and the system generates department pages based on that.

Pages in Individual Windows

- Task pages – This page type contains all tasks that are part of a business process that an end user is likely to need to complete. On the task page, the Action Pane promotes the commands that are most frequently used within a given context. Task pages are very similar to card forms in the Classic client.
  - Cards — Pages with one or more FastTabs, or sections of a page that can be expanded or collapsed. Cards contain details and statistics about a single entity, such as a customer or item.
  - Documents — Documents with one or more FastTabs for the document header information and one FastTab for the document lines. For example, sales orders and credit memos are document task pages that use FastTabs.
  - Journals/worksheets — Lists where lines are created as a draft and then acted upon, often with specific details shown under the list, such as Sales Price Worksheet or Item Journal.
  - List task pages — An automatic reuse of the page definition for a list place, which is used to render the page as a pop-up list in View mode, in Edit mode, or in Selection mode where a value is brought back to the calling window. The list task page has one list and no FastTabs with fields.
  - List plus — Pages that include one FastTab with a list and additional FastTabs with fields or more lists. This page does not represent a business entity but is used for custom display or processing of information from multiple entities, such as the application worksheet or some statistics forms.
  - Matrices — Pages with a grid where each cell, not a row, represents an entity or a sum of entities.
  - Wizards — Sets of related pages that guide users through a task.
- Dialog boxes – This page type is used to briefly show pages that ask the user for parameters for further processing. These pages do not represent an entity.

- Confirmation dialog boxes — Pages that ask users for input for further processing by posing a question. Generally, the page includes a details section with information relevant to the decision.
- Request forms — Pages with one or more FastTabs each containing a filter pane to filter a specific type of entity. In addition, the page can contain a FastTab at the top with user options and preferences for how the processing should be done. These pages are autogenerated based on the definitions in the reports and batch jobs.

Component Page Types Embedded in Other Page Types

- FactBoxes — This page type is similar to a mini-card that provides supplemental information. A FactBox takes as input a key value from the hosting page, which it uses to look up and display information that is related to the key value. For example, the Customer Statistics FactBox on a sales order takes a customer number as input and uses it to display the customer's balance, outstanding amount, and other information.
- Role Center parts — This page type is similar to FactBoxes, except that Role Center parts do not take input for queries. They always use the same query and show a constant number of items on a list. For example, a top 10 list of items is a Role Center part. The top 10 items may change over time, but the list of 10 items is fixed.

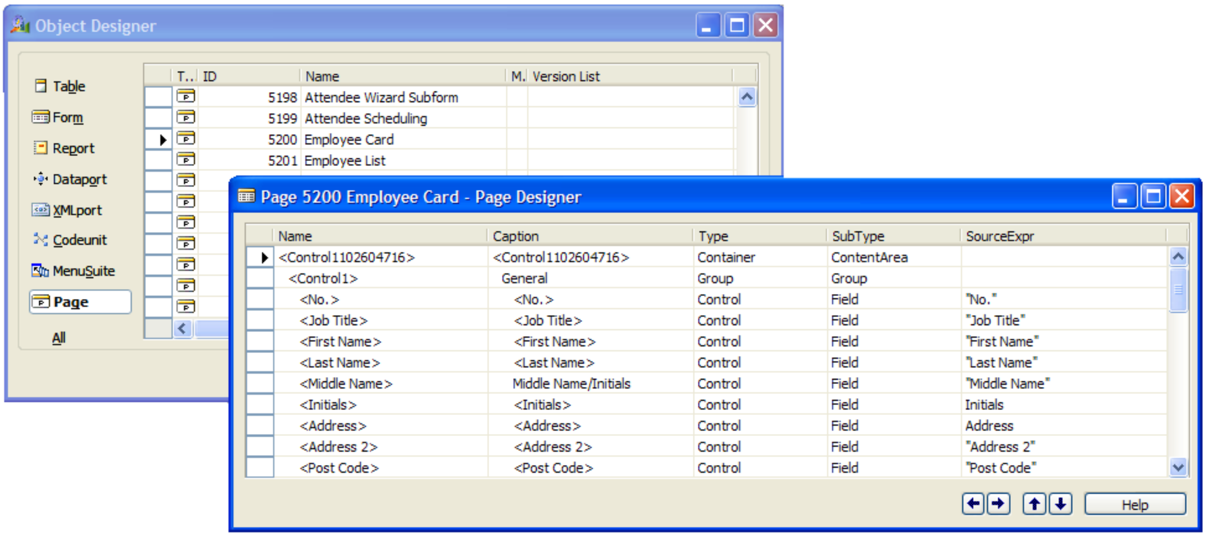
Designing Pages with Page Designer

With the new page layouts, you can create pages that focus user attention on the data and actions that are most frequently used. This creates a more effective, less confusing user experience, and new users learn to use the page more quickly.

First, think about the types of users that will use the new window, and list the tasks they will do. Make a list of the fields, commands and links needed for all flavors of the task to the window. Pick the page types that best match the type of content that you listed for the page. The source table for the page should be the table that contains the primary set of data that the page will display.

Next, consider if any supplemental sources of information will add value to the page and add these as FactBoxes. It is possible that existing FactBoxes could support your design.

Then, once you have decided what the design of a page should be; create it using Page Designer. When you design pages, you specify a hierarchy of page elements. You determine the details of how data is displayed by adjusting the properties of each element on the page.



T..	ID	Name	M.	Version List
P	5198	Attendee Wizard Subform		
P	5199	Attendee Scheduling		
P	5200	Employee Card		
P	5201	Employee List		

Name	Caption	Type	SubType	SourceExpr
<Control1102604716>	<Control1102604716>	Container	ContentArea	
<Control1>	General	Group	Group	
<No.>	<No.>	Control	Field	"No."
<Job Title>	<Job Title>	Control	Field	"Job Title"
<First Name>	<First Name>	Control	Field	"First Name"
<Last Name>	<Last Name>	Control	Field	"Last Name"
<Middle Name>	Middle Name/Initials	Control	Field	"Middle Name"
<Initials>	<Initials>	Control	Field	Initials
<Address>	<Address>	Control	Field	Address
<Address 2>	<Address 2>	Control	Field	"Address 2"
<Post Code>	<Post Code>	Control	Field	"Post Code"

By only specifying the content that should be shown and how it should be presented, you do not need to provide a static layout and do pixel-level editing. The central activity in page design is specifying the contents of the page in relation to the different Microsoft Dynamics NAV page types.

Then, simplify the user experience by reducing what users see by default:

- Promote commands to the Action Pane that users will use frequently. Create groups of commands in the Action Pane that suggest different tasks or the state at which the commands are used. In each group, provide a sequence for commands to suggest sequence of use. Use large icons for the default commands and small icons for alternative commands and less frequently used ones.
- In every FastTab, set the **Importance** property to **Additional** for all fields that users will use less than three out of ten times. Set the property to **Promoted** for one to four important fields so that users can see their values in the FastTab summary without opening the FastTab.
- Display one to three FactBoxes by default. This is usually one to two FactBoxes and Notes.

Finally, add the new page to the menu suite so that it will be included on the Departments place. If the page you created is a list place, then consider if a specific user profiles should get a link to the page in their role-oriented navigation pane. If the new page is a task page, then consider which list places from which it would be useful to link to and add the links.

End users can customize the appearance of the pages that are available to them. For example, in list pages, they can choose to hide or show both columns and specify which FactBoxes should be displayed. End users can also change the display order of FactBoxes or choose to hide nonrelevant FactBoxes for their Role Center or a particular page.

The personalization features for end users do not allow them to access any parts of the system to which they have not been given permission. However, it does let users hide page elements that they would prefer not to see or show elements that may not be shown by default. When you are designing a page, you should include a comprehensive list of page elements and note that the most common changes that a user makes is to remove elements from rather than add elements to the page that you are designing.

For a step-by-step demonstration of how to create and modify pages in Microsoft Dynamics NAV “6.0”, see Step-by-Step Walkthrough: Creating and Modifying Pages.

Reports

Reports for the RoleTailored client are based on the SQL Server Reporting Services format and introduce improvements over the existing reporting engine in C/SIDE. Features of the new reports include:

- Graphical and color capabilities, such as pictures, charts, graphs, and custom objects.
- Interactive features that allow users to:
    - Expand or contract to show or hide line details.
    - Dynamically change sorting on single or multiple columns.
- Rich calculation library.
- Export to Microsoft Office Excel® and PDF functionality.

The extra information that the RoleTailored client needs is stored in the report definition language (RDL) format and in extra properties in the report object. RDL is a SQL Server Reporting standard format for reports. There are a number of available report editors, such as Microsoft Visual Web Developer™ Express Edition, which is available as a free download.

To create a report for the RoleTailored client, you define the data items for the report and then export the report to a SQL Server Reporting Services project in Visual Studio. You use SQL Server Reporting Services to design the layout of the report and any postprocessing, such as grouping and totaling. You migrate existing reports from previous versions of Microsoft Dynamics NAV or from the Classic client and design new reports in the same way.

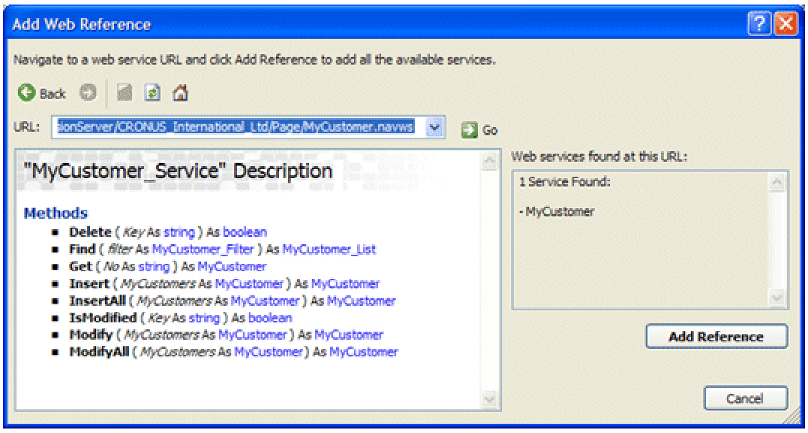
Some properties from Classic reports are not supported in the new architecture. In general, these properties deal with report layout, totaling, grouping, and printer specifications. When Visual Studio is opened, these properties are ignored and need to be applied from the layout designer. This only needs to be done once, and any changes made to reports in Visual Studio are saved when you save your report.

For a step-by-step demonstration of how to create reports for Microsoft Dynamics NAV “6.0”, see Step-by-Step Walkthrough: Creating a Report.

Web Services

Page-Based Web Services

When a page is published as a Web service, a fixed set of eight methods is exposed to developers so that they can manage common record handling operations, such as Insert, Modify, and Find.



Navigate to a web service URL and click Add Reference to add all the available services.

URL: ionServer/CRONUS_International_Ltd/Page/MyCustomer.navws

"MyCustomer_Service" Description

Methods

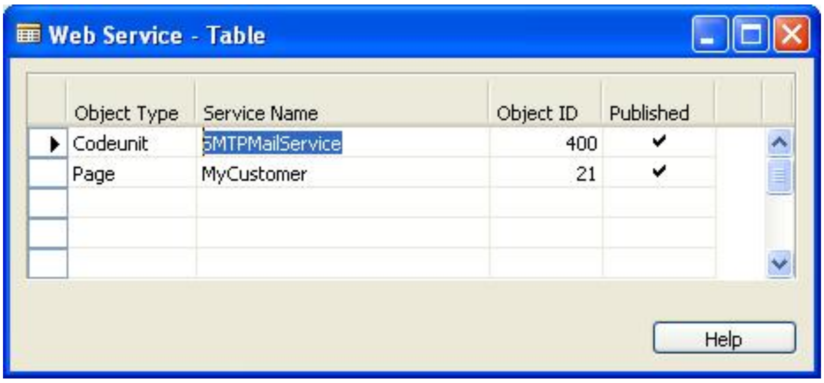
- **Delete** (*Key* As string) As boolean
- **Find** (*filter* As MyCustomer_Filter) As MyCustomer_List
- **Get** (*No* As string) As MyCustomer
- **Insert** (*MyCustomers* As MyCustomer) As MyCustomer
- **InsertAll** (*MyCustomers* As MyCustomer) As MyCustomer
- **IsModified** (*Key* As string) As boolean
- **Modify** (*MyCustomers* As MyCustomer) As MyCustomer
- **ModifyAll** (*MyCustomers* As MyCustomer) As MyCustomer

Web services found at this URL:

1 Service Found:

- MyCustomer

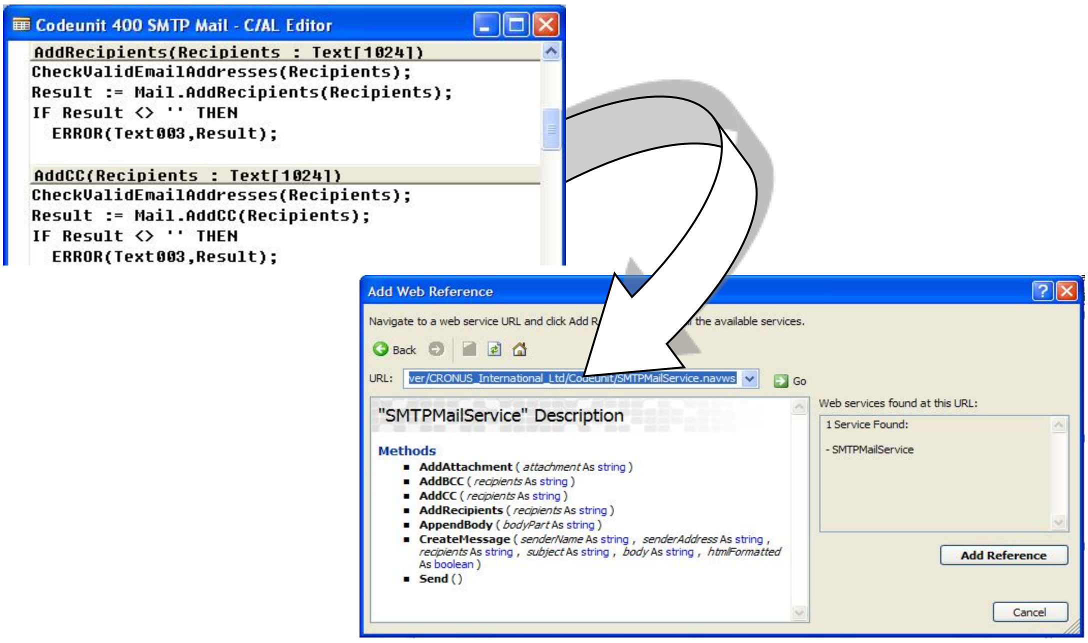
To make a simple page-based Web service in Microsoft Dynamics NAV “6.0”, you write a page object and then add that object to the Web Service table. The following figure demonstrates a page-based Web service called MyCustomer.



Object Type	Service Name	Object ID	Published
Codeunit	SMTPMailService	400	✓
Page	MyCustomer	21	✓

Codeunit-Based Web Services

When a codeunit is published as a Web service, any methods that you have defined on the codeunit are exposed to developers so that they can integrate with the codeunit.



Codeunit 400 SMTP Mail - C/AL Editor

```
AddRecipients(Recipients : Text[1024])
CheckValidEmailAddresses(Recipients);
Result := Mail.AddRecipients(Recipients);
IF Result <> '' THEN
  ERROR(Text003,Result);

AddCC(Recipients : Text[1024])
CheckValidEmailAddresses(Recipients);
Result := Mail.AddCC(Recipients);
IF Result <> '' THEN
  ERROR(Text003,Result);
```

Add Web Reference
Navigate to a web service URL and click Add R
the available services.
Back

URL: ver/CRONUS_International_Ltd/Codeunit/SMTPMailService.navws

"SMTPMailService" Description

Methods

- **AddAttachment** (*attachment* As string)
- **AddBCC** (*recipients* As string)
- **AddCC** (*recipients* As string)
- **AddRecipients** (*recipients* As string)
- **AppendBody** (*bodyPart* As string)
- **CreateMessage** (*senderName* As string , *senderAddress* As string , *recipients* As string , *subject* As string , *body* As string , *htmlFormatted* As boolean)
- **Send** ( )

Web services found at this URL:

1 Service Found:

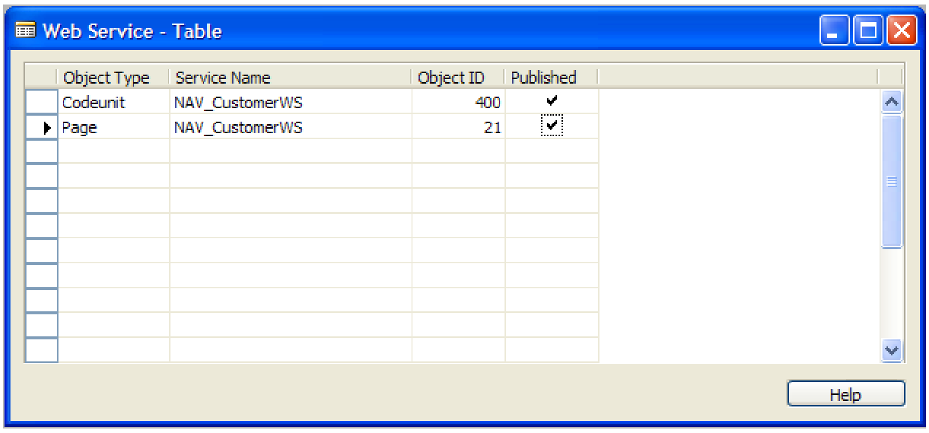
- SMTPMailService

To make a simple codeunit-based Web service in Microsoft Dynamics NAV “6.0”, you write a codeunit object and then add that object to the Web Service table.

Page and Codeunit–Based Web Services

On a page object, you can add extra methods to the Web service by associating codeunits with it. When a page and codeunit are published as a Web service, a fixed set of eight methods on the page and any additional methods are exposed to developers.

To make a page and codeunit–based Web service, you write a page object and codeunit and add them to the Web Service table. You then associate the two objects by entering the same service name for each object.



Object Type	Service Name	Object ID	Published
Codeunit	NAV_CustomerWS	400	✓
Page	NAV_CustomerWS	21	☑

Codeunits That Use Complex Types in Web Services

You can develop Web services that have complex parameters. Complex parameters are different from simple parameters, such as integer or Boolean, because they can have their own structure. A sales order is an example of a complex object, because it has one header and multiple lines, each of which has its own details.

You pass complex information to a Web service by using an XMLport as a parameter to codeunit functions. This approach can be best when you do not want to expose all fields from a page object. Instead, you may want to publish a Web service that offers access to a subset of fields on a page. To do this, you write an XMLport that captures the required field and make that XMLport to be a parameter to a function in a codeunit. You then publish the codeunit with a page. From the Web service interface, you can then access a function with the XMLport as a parameter. Getting the values from C/AL is still possible, because you can access the business logic when the validate triggers are called.

Functionality Redesign Needs in Microsoft Dynamics NAV "6.0"

Some functionality in Microsoft Dynamics NAV "6.0" must be redesigned to work with the three-tier architecture, which includes the RoleTailored client and Microsoft Dynamics NAV Server.

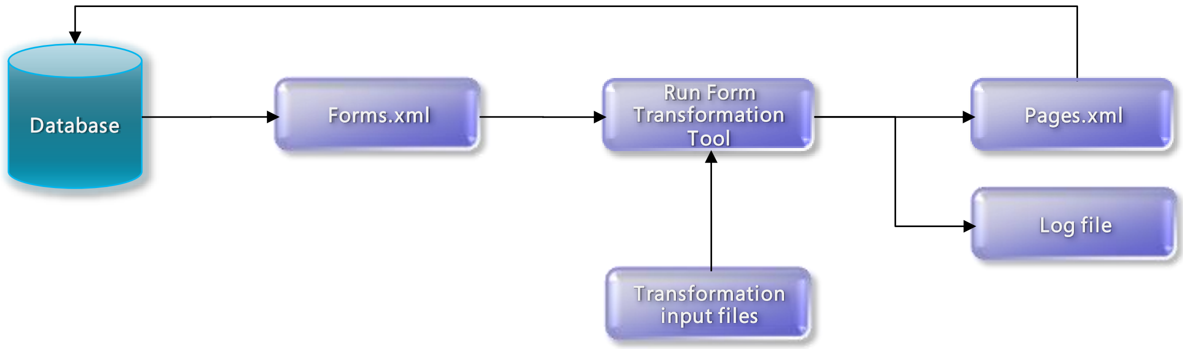
Removing Unsupported Functions, Triggers, and Properties

Functions, triggers, or properties that are not supported in the service architecture will not cause errors but instead run with no result. The Form Transformation Tool removes triggers and properties that are not supported in pages and keeps track of these removals in a log file.

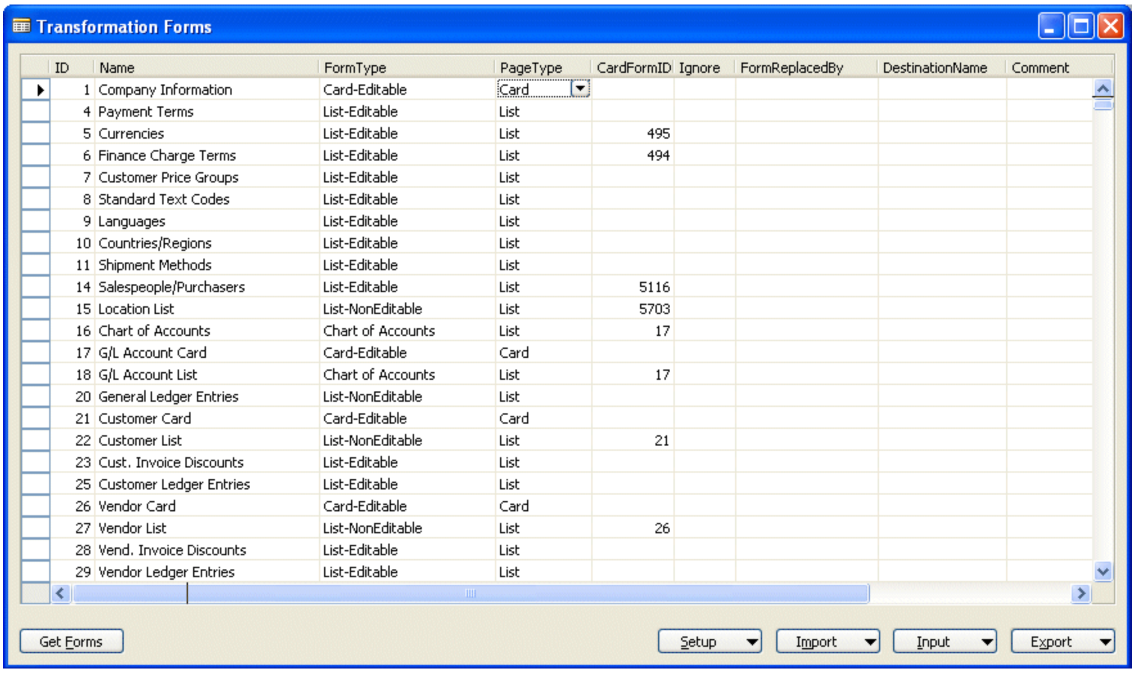
Migrating Forms to Pages with the Form Transformation Tool

To help developers migrate forms to pages, Microsoft Dynamics NAV "6.0" includes the Form Transformation Tool. It is a mapping tool that you use to generate pages based on forms in your current application. The tool is based on structured rules that have been developed for mapping forms to pages. The tool generates new pages with controls and properties that correspond to particular new behavior available in the RoleTailored client. Most new page objects that are created contain the same functionality as the original forms.

The Form Transformation Tool creates a new page object for each form in your application by analyzing the controls and properties on each form and creating a page with the corresponding controls and properties. It does not change or delete the existing forms during the transformation process.



You first map every form in the application to one of the new page types that have been defined for the three-tier architecture. For every form that you want to transform to a page, you must specify if the form is a card type, list type, document type, FactBox type, and so on.



ID	Name	FormType	PageType	CardFormID	Ignore	FormReplacedBy	DestinationName	Comment
1	Company Information	Card-Editable	Card					
4	Payment Terms	List-Editable	List					
5	Currencies	List-Editable	List	495				
6	Finance Charge Terms	List-Editable	List	494				
7	Customer Price Groups	List-Editable	List					
8	Standard Text Codes	List-Editable	List					
9	Languages	List-Editable	List					
10	Countries/Regions	List-Editable	List					
11	Shipment Methods	List-Editable	List					
14	Salespeople/Purchasers	List-Editable	List	5116				
15	Location List	List-NonEditable	List	5703				
16	Chart of Accounts	Chart of Accounts	List	17				
17	G/L Account Card	Card-Editable	Card					
18	G/L Account List	Chart of Accounts	List	17				
20	General Ledger Entries	List-NonEditable	List					
21	Customer Card	Card-Editable	Card					
22	Customer List	List-NonEditable	List	21				
23	Cust. Invoice Discounts	List-Editable	List					
25	Customer Ledger Entries	List-Editable	List					
26	Vendor Card	Card-Editable	Card					
27	Vendor List	List-NonEditable	List	26				
28	Vend. Invoice Discounts	List-Editable	List					
29	Vendor Ledger Entries	List-Editable	List					

You then run the Form Transformation Tool over your forms. You can specify which forms are transformed and which forms are ignored. The Form Transformation Tool never deletes code when it generates a page from a form. However, if a form has a trigger that contains code and no corresponding trigger exists on a page, the code is either moved to a similar trigger or to the log file.

The Form Transformation Tool never deletes code when it generates a page from a form. However, if a form has a trigger that contains code and no corresponding trigger exists on a Page, the code is either moved to a similar trigger or to the log file. This information is useful when you look at the untransformed forms and decide whether they should be transformed. These forms must be redesigned before they can be transformed successfully. Otherwise, you can ignore some or all of these forms and create new pages that contain the same functionality in Page Designer.

The success of the Form Transformation Tool depends on the extent and type of customizations that you have done on a form. If the customizations correspond to standard application design, then the transformation process is easier. If the customizations are more complex, then there may be problems in the transformation process.

Also, some forms in Microsoft Dynamics NAV “6.0” have been changed to ensure that pages that contain the same functionality can be easily generated by the Form Transformation Tool. You may need to customize a subset of forms again to ensure that you can successfully transform these pages.

The following sections describe some specific changes to how types of forms are handled or have changed in Microsoft Dynamics NAV “6.0”.

Statistics Forms

Statistics forms that contain more than two columns must be adjusted before they can be transformed correctly. To transform to a page, the columns must be grouped in a matrix, and the all cells in the page must be filled. You must add a placeholder control to empty cells to fill the space.

It may be more useful to redesign some statistics forms as reports, because statistics forms are generally used for data analysis.

Forms with Information Frames or Filter Frames

Forms that contain information frames, such as the Item Charge Assignment or Cash Receipt Journal form, or forms with filter frames, such as the Sales Price form, must be redesigned in the same way as statistics forms. You must place a frame control around the Info Frame section to successfully transform it to a page.

If it is not redesigned, then the default two-column layout of the RoleTailored client is used. In some cases, you may want to rearrange the fields to get a more user-friendly field order in the RoleTailored client.

Matrix Forms

The RoleTailored client does not support matrix controls. You can map a matrix control on a form to a grid control on a page, but there are the following limitations to this approach.

- You must make a new request page that will be used to define the filters and selections that should be applied and a button to show the result in the grid.
- The grid is read-only, and you cannot make grids or matrix forms in the RoleTailored client that are used for data entry.
- The grid contains a finite number of columns. The default value can be changed if you need a larger grid but will always be static for that page.

Journals and Worksheets with Batches

Journals and worksheets with batches are now accessed through a list place of batches, so you must add an Edit Journal command to the batch form.

Navigation panes on Role Centers should refer to list places.

Type-Specific Filtering on Forms

The RoleTailored client does not support filtering in the Department place (also known as the MenuSuite). For example, in the Classic client, you can apply a filter to the sales list for quotes. From the Menu Suite, you can then open a list of only quotes. The RoleTailored client does not have this functionality, so you must make a new page for each filter value. In this example, the sales list becomes six new lists that are specific to the different document types.

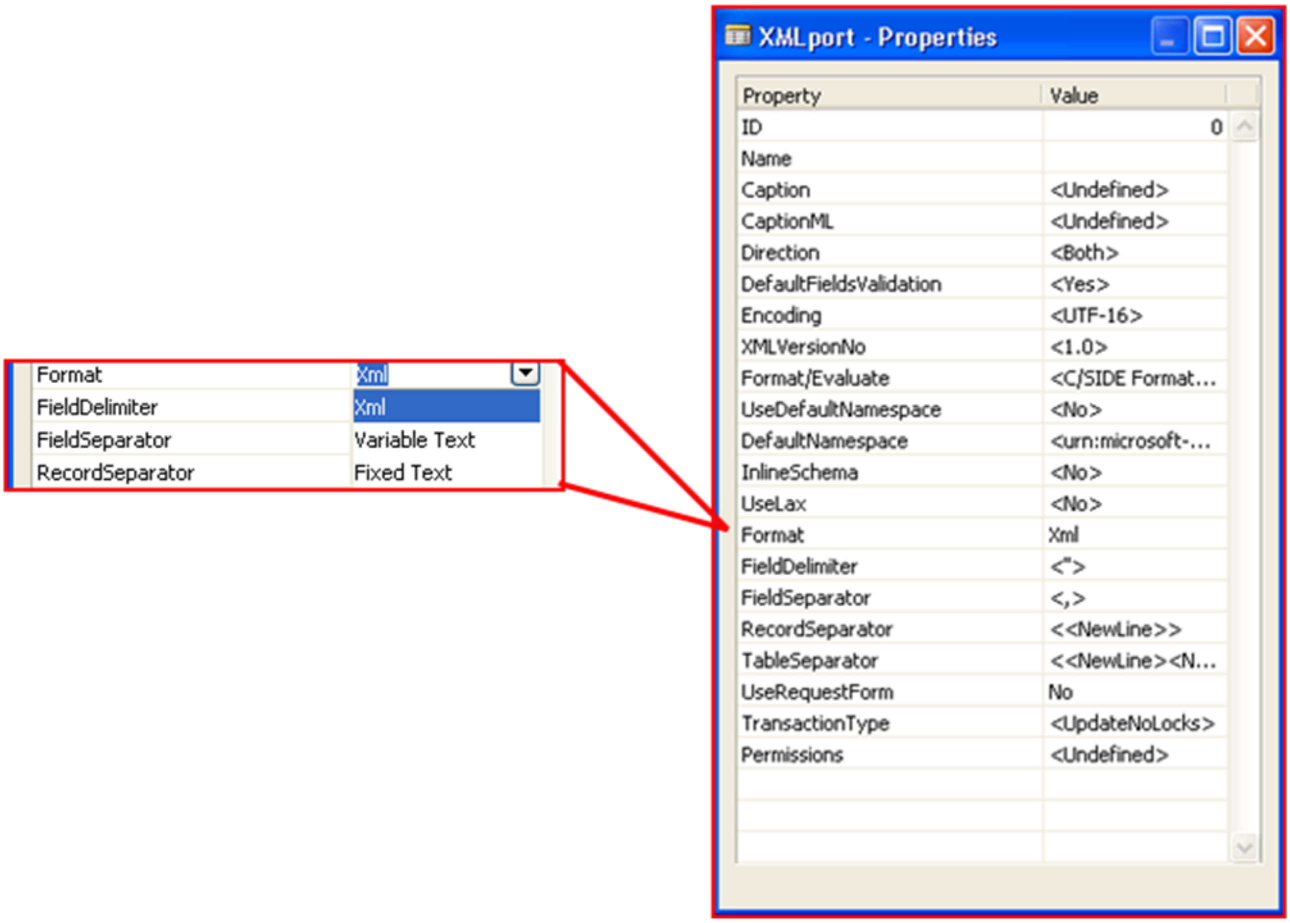
Understanding XMLport and Dataport Usage

In previous versions of Microsoft Dynamics NAV and in the Classic client, you can use XMLports and dataports to import and export XML documents and text files (comma separated, character delimited, and fixed width).

In the RoleTailored client and Microsoft Dynamics NAV Server, the two object types have been merged into one object, the XMLport, which provides the same functionality.

You develop XMLports with the C/SIDE XMLport Designer for both architectures. When you are developing for Microsoft Dynamics NAV Server, you can use additional properties and have access to additional functionality. Request pages and import and export text functionality have been added to XMLports. These properties are visible in the Classic client so that you can access them in XMLport Designer. However, setting or executing an XMLport with these properties in the Classic client has no effect.

The following figure shows the additions to XMLport Designer:



Format	Xml
FieldDelimiter	Xml
FieldSeparator	Variable Text
RecordSeparator	Fixed Text

Property	Value
ID	0
Name	
Caption	<Undefined>
CaptionML	<Undefined>
Direction	<Both>
DefaultFieldsValidation	<Yes>
Encoding	<UTF-16>
XMLVersionNo	<1.0>
Format/Evaluate	<C/SIDE Format...
UseDefaultNamespace	<No>
DefaultNamespace	<urn:microsoft-...
InlineSchema	<No>
UseLax	<No>
Format	Xml
FieldDelimiter	<">
FieldSeparator	<,>
RecordSeparator	<<NewLine>>
TableSeparator	<<NewLine><N...
UseRequestForm	No
TransactionType	<UpdateNoLocks>
Permissions	<Undefined>

Importing and exporting text also works differently in XMLports:

- XMLports do not have a OnBeforeExportRecord trigger.
- XMLports support multilanguage for the CaptionML and Language properties.
- XMLports support UTF-8 and UTF-16 formats for XML.

Managing File Upload and Download Locations

When you create files with the three-tier architecture, you must update your code so that files are created and managed in the correct location. Files must be downloaded, processed, and uploaded by the middle tier that is running Microsoft Dynamics NAV Server.

This guidance also applies to processing-only reports, because they are commonly saved to file or printed to file after they have been run.

The following new functions have been created to make this file transfer and management easier:

- FILE.UPLOAD and FILE.UPLOADINTOSTREAM — Sends a file from the RoleTailored client to the middle tier.
- FILE.DOWNLOAD and FILE.DOWNLOADINFROMSTREAM — Sends a file from the middle tier to the RoleTailored client.

Assessing Automation Objects

Code for automation objects runs on Microsoft Dynamics NAV Server. Therefore, you must ensure that files are generated on the computer that is running Microsoft Dynamics NAV Server and that code is running in the correct places for the automation to run successfully. The following scenarios demonstrate how you would need to change your automation objects to support Microsoft Dynamics NAV Server.

> **Note** You only need to install COM objects on the computer that is running Microsoft Dynamics NAV Server. You do not need to install COM objects on every client.

To Use a COM Component to Generate an Excel File for a User

With C/AL code running on Microsoft Dynamics NAV Server, you use the CREATE function to create the automation object and then run its SAVE function to save it to that computer. When the RoleTailored client needs access to the file, you use the FILE.DOWNLOAD function to send it from the computer that is running Microsoft Dynamics NAV Server to the client.

To ensure that this works correctly, the COM DLL must be stored and registered on the computer that is running Microsoft Dynamics NAV Server.

To Use a COM Component to Carry Out External Processing, Such as Call a Web Service

With C/AL code running on Microsoft Dynamics NAV Server, you use the CREATE function to create an automation object and then call a method on the object. When the method is run, any return value is passed back to the code.

To ensure that this works correctly, the COM DLL must be stored and registered on the computer that is running Microsoft Dynamics NAV Server.

To Use a COM Component to Read from an MSMQ Message

With C/AL code running on a computer that is running Microsoft Dynamics NAV Server, you use the CREATE function to create an automation object and the READ function to read it from the message queue. For the RoleTailored client, the Microsoft Dynamics NAV Server processes the result of the READ function.

> **Note** If the queue location is specifically locally, such as **./myqueue**, the code reads from a message queue called **myqueue** on the computer that is running Microsoft Dynamics NAV Server and not from a message queue on the client.

To Use a Visual OCX with Its Own Window

You cannot use an OCX that has its own window to perform special tasks for the client.

In the three-tier architecture, the client cannot run code or interact with other components that are also running on the client or with the end user. To duplicate the behavior of the existing client, the other applications that run on the client must be redesigned as Web service–based solutions. The client application can then call the Web service, which then delivers the information that is required by the system. You can also write code to provide actions to a page that reads from the database after the Web service has been called.

To Understand Client Interaction with Hardware

The RoleTailored client cannot interact with local hardware, such as barcode scanners and electronic scales.

In the three-tier architecture, business logic code is run on Microsoft Dynamics NAV Server instead of on the client. Any existing components that require business logic to run on the client must be redesigned. Possible redesigns include using Web services to transmit data to Microsoft Dynamics NAV Server or writing a client component that writes values directly to the SQL Server database using C/FRONT.NET.

Understanding Display Property Behavior in Forms and Pages

None of the properties that govern fonts, such as **FontSize** and **FontItalic**, are supported in the three-tier architecture. These properties are ignored by the RoleTailored client because the behavior of the UI is controlled by the client.

The **Visible**, **Editable**, and **Enabled** properties have also been changed. In forms that target the Classic client, you can set the Boolean value of the property from the code. In pages that target the RoleTailored client, the value is defined in a property of the control. it is also possible to set the value from an expression. For example, you can set the value to **true** if a certain value is present in a variable or another control has a particular value.

Managing Style Sheets

Style sheets require different templates for the Classic client and the RoleTailored client. You need to remake existing style sheets for the RoleTailored Client, but they are still stored in the same table in Microsoft Dynamics NAV.

When you store a style sheet, you specify that a particular style sheet is for a form (for the Classic client) or a page (for the RoleTailored client) in an additional field.

Managing Infinitely Long-Running Processes

Microsoft Dynamics NAV Server is not designed to host processes that run indefinitely. An example of an infinitely long process is a solution that runs on the Dynamics NAV Application Server and listens for and processes external COM events.

To build a similar solution, you must use a more event-driven approach instead of a polling or always-on approach. For example, you can implement a solution that uses a Web service as the event trigger. This also increases interoperability and reliability.

Multiple Client Support on Microsoft Dynamics NAV "6.0"

The Microsoft Dynamics NAV Classic client and Microsoft Dynamics NAV application server can run on the same SQL Server database that is used by the NAV Server. This means that you can run both RoleTailored clients and Classic clients in one system, to offer different users in an organization access to the same application and data.

The biggest benefit of being able to deploy different clients in one system is that specific solutions that are not immediately migrated to support the three-tier architecture can still be used in an organization. Also, any integration solutions that run on the Microsoft Dynamics NAV application server continue to work.

Deploying different client versions does introduce constraints in Microsoft Dynamics NAV. The main difficulty in supporting multiple clients is the increased complexity that is required when developing the solution. In the two-tier architecture with the Classic client, the client runs all code in its local environment. In the three-tier architecture with the RoleTailored client, all code runs on the computer that is running Microsoft Dynamics NAV Server. When you are making design decisions, you must ensure parity in code execution behavior between the two clients. If differences arise, then different clients may perform the same task but arrive at a different final result, which can result in data consistency issues.

Writing Code That Targets Multiple Clients

Most code behaves the same on the Classic client or the RoleTailored client. However, the behavior is different when running forms and pages.

For example, you can write C/AL code to run a form as follows:

```
FORM.RUN(<form id>,<optional parameters>);
```

This code is interpreted at run time in the following ways:

- If it is on the computer that is running on the Classic client, then it will run the form object with that ID.
- If it is on the computer that is running Microsoft Dynamics NAV Server, then it will run the page object with that ID.

This has two implications when you are designing code that targets multiple clients:

- Forms and pages must stay synchronized. If your codeunit calls the Customer form, then the Customer page ID must have the same ID as the Customer form. Otherwise, it will not work.
- Forms and pages must have the same methods with the same number of parameters. Otherwise, you need to write additional code to support forms and pages.

To allow you to write code that targets either the Classic client or the RoleTailored client, a new system property called ISSERVICETIER has been created. If code is being executed on the service, then it returns TRUE. Otherwise, it returns FALSE. You can then use ISSERVICETIER to write code that is explicit about whether it is run on the service.

> **Note** This property is similar to the GUIALLOWED system property that was introduced to help with writing code for the Microsoft Dynamics NAV Application Server.

For example, if your form object has the method WRITE with two integer parameters, then the equivalent page object must also have the same method with the same parameters. If the methods do not have the same signature, then any code that is outside the object that calls the page or form will fail. In the following example, the page object has an extra *text* parameter. You can write the following code to interpret which client is running:

```
IF (ISSERVICETIER) THEN
  FORM.WRITE(int,int,text)
ELSE
  FORM.WRITE(int,int)
ENDIF
```

This approach does introduce some complications for writing and maintaining the code:

- It reduces how easy the code is to read, understand, and maintain. You may need to maintain and keep code synchronized in multiple places. Reports and code that use automation objects and files are different between the two platforms. For example, Office integration style sheets function differently in the Classic client and RoleTailored client. Writing and maintaining code for style sheets requires work in two formats and understanding for how particular style sheets should be applied to the different clients.
- It increasing testing needs, because code will need to be tested with multiple clients to ensure that it functions correctly.
- Both clients use the MenuSuite object for navigation, but the design is implemented differently. In the Classic client, a user opens a card and then can jump to a list. In the RoleTailored client, a user is presented with a list and can then select which page (card) to open. Supporting both design types in Microsoft Dynamics NAV increases the work to understand and maintain changes.
- Both clients have different implementations of the feature that is used to record miscellaneous notes or comments about a document or contact. The Classic client uses *comments*, and the RoleTailored client uses *notes*. Both features provide similar functionality, but different users are not able to see the information that is entered by another user who uses a different client.

Writing Reports That Target Multiple Clients

You can build a single report object that can run on both the Classic client and RoleTailored client. The most significant change is that the layout of the reports is defined differently. When you design a report for the Classic client, you design the layout in Section Designer. When you design a report for the RoleTailored client, you design the layout in an RDL editor, such as Visual Studio Report Editor.

Some properties from Classic reports are not supported in the new architecture. In general, these properties deal with report layout, totaling, grouping, and printer specifications. When Visual Studio is opened, these properties are ignored, and you need to apply them from the layout designer. You only need to do this once, and any changes made to reports in Visual Studio are saved when you save your report.

You should also move any special code that you have written for a report from the section triggers to a data item section in the report or to a codeunit.

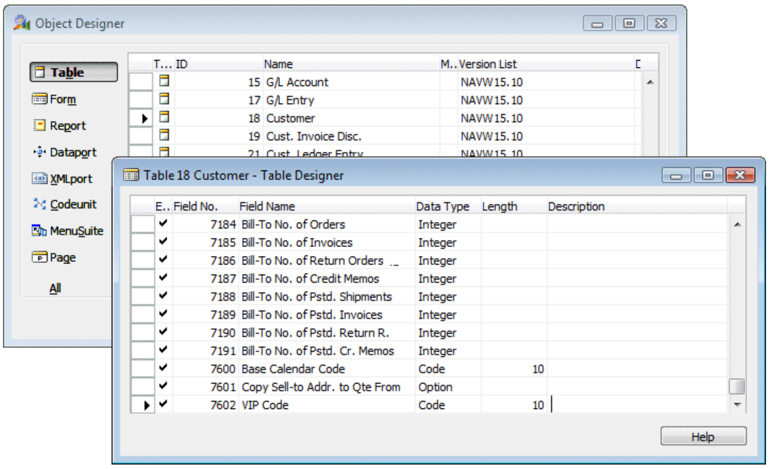
Step-by-Step Walkthrough: Creating and Modifying Pages

In this walkthrough, you create and modify pages in the RoleTailored client. You start by adding a new field to the **Customer** table and then change the **Customer List** place and the **Customer Card** page before they can display this new field. You finally create a new FactBox that displays some details about the customer that is currently selected in the **Customer List** place.

To Add a New Field to the Customer Table

For this walkthrough, you have created a new table called **VIP**. This table contains the very important person (VIP) status settings (Bronze, Silver, Gold, and Diamond) that you can assign to your customers.

1. Open the Classic client and use Table Designer to extend the **Customer** table by adding a field called **VIP Code**. This field contains information about each customer's VIP status.
2. Link the **VIP Code** field to the **VIP Code** table.



T...	ID	Name	M..	Version List
	15	G/L Account		NAVW15.10
	17	G/L Entry		NAVW15.10
	18	Customer		NAVW15.10
	19	Cust. Invoice Disc.		NAVW15.10
	21	Cust. Ledger Entry		NAVW15.10

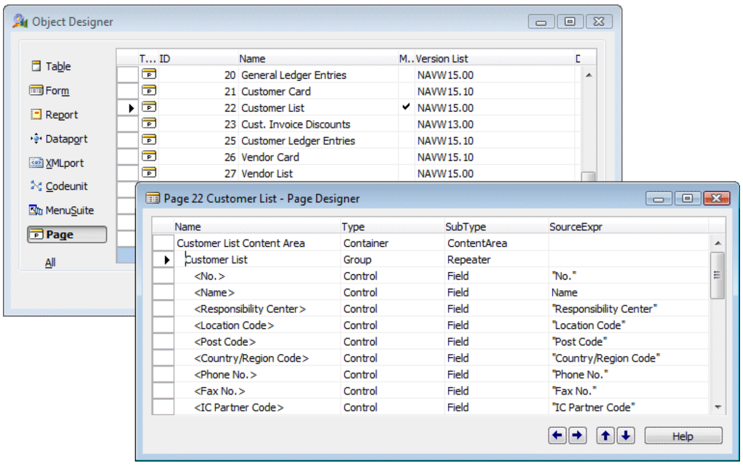
Table 18 Customer - Table Designer

E..	Field No.	Field Name	Data Type	Length	Description
✓	7184	Bill-To No. of Orders	Integer		
✓	7185	Bill-To No. of Invoices	Integer		
✓	7186	Bill-To No. of Return Orders	Integer		
✓	7187	Bill-To No. of Credit Memos	Integer		
✓	7188	Bill-To No. of Pstd. Shipments	Integer		
✓	7189	Bill-To No. of Pstd. Invoices	Integer		
✓	7190	Bill-To No. of Pstd. Return R.	Integer		
✓	7191	Bill-To No. of Pstd. Cr. Memos	Integer		
✓	7600	Base Calendar Code	Code	10	
✓	7601	Copy Sell-to Addr. to Qte From	Option		
✓	7602	VIP Code	Code	10	

To Modify the Customer List Page

You then add the information about the new **VIP Code** field to the **Customer List** page.

1. In Object Designer, open the **Customer List** page in Page Designer.



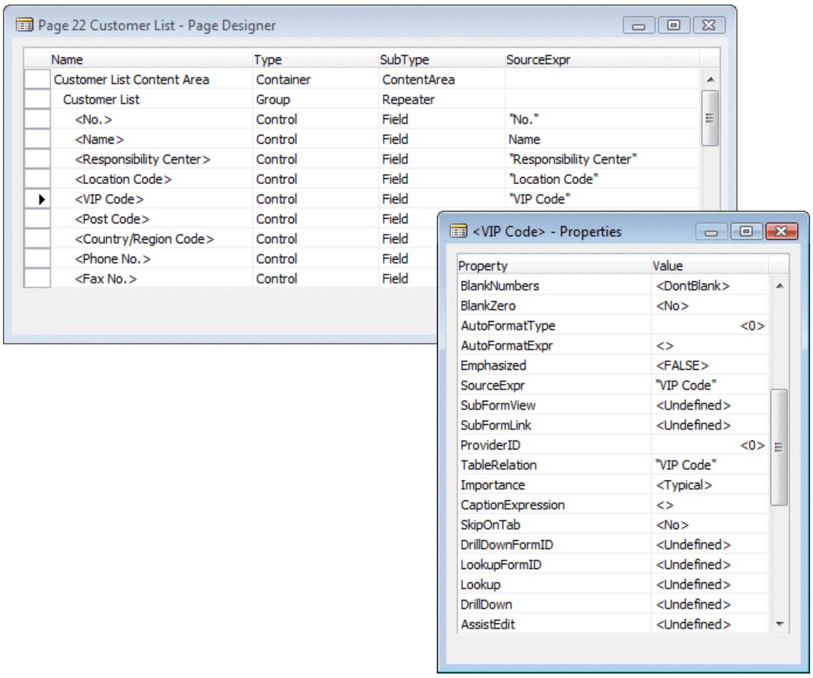
T...	ID	Name	M..	Version List
	20	General Ledger Entries		NAVW15.00
	21	Customer Card		NAVW15.10
	22	Customer List	✓	NAVW15.00
	23	Cust. Invoice Discounts		NAVW13.00
	25	Customer Ledger Entries		NAVW15.10
	26	Vendor Card		NAVW15.10
	27	Vendor List		NAVW15.00

Page 22 Customer List - Page Designer

Name	Type	SubType	SourceExpr
Customer List Content Area	Container	ContentArea	
Customer List	Group	Repeater	
<No.>	Control	Field	"No."
<Name>	Control	Field	Name
<Responsibility Center>	Control	Field	"Responsibility Center"
<Location Code>	Control	Field	"Location Code"
<Post Code>	Control	Field	"Post Code"
<Country/Region Code>	Control	Field	"Country/Region Code"
<Phone No.>	Control	Field	"Phone No."
<Fax No.>	Control	Field	"Fax No."
<IC Partner Code>	Control	Field	"IC Partner Code"

Pages have a hierarchical structure that allows you to organize the contents of the page into groups called *containers*. The first element or line in Page Designer defines the page's primary group. In this example, it is called **Customer List Content Area**. This element has the subtype **ContentArea**, which means that the contents of this group are displayed in the content area when it is displayed in the RoleTailored client. Other important subtypes are InfoParts and HomeParts. These subtypes are used later in this walkthrough.

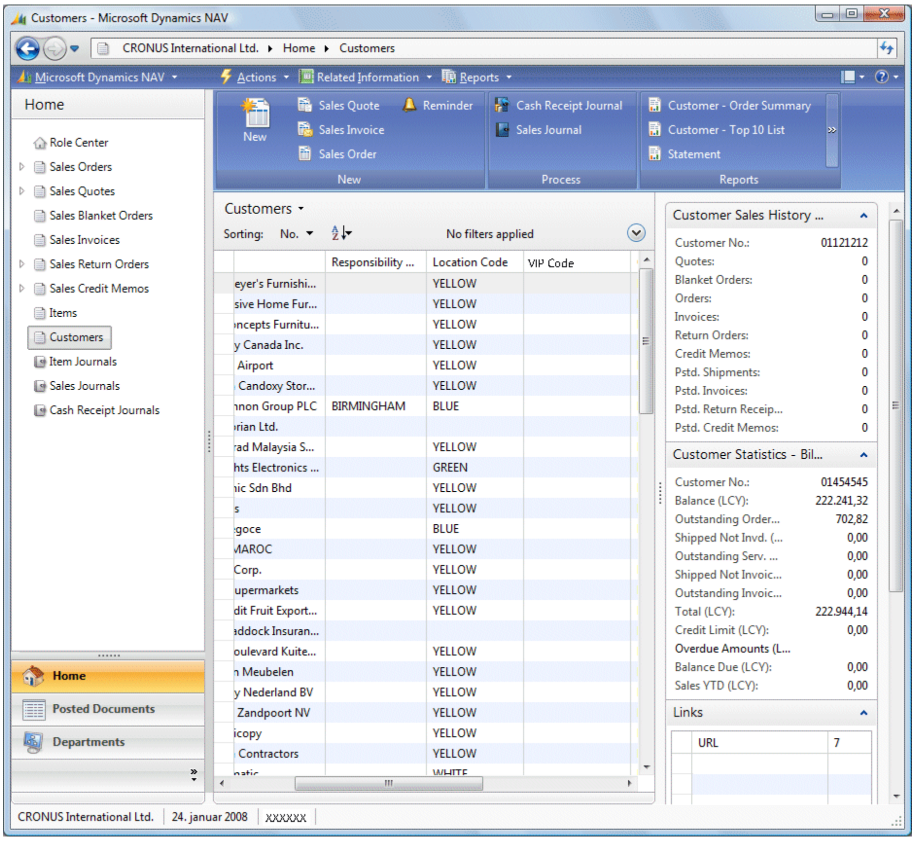
The hierarchical structure of the page is determined by the indentation of the rows that are shown in Page Designer. **Customer List** has elements that are indented below it and is a separate group within the **Customer List Content Area** container. In this example, the **Customer List** group contains the fields that will appear as column headers in the **Customer List** page.



Name	Type	SubType	SourceExpr
Customer List Content Area	Container	ContentArea	
Customer List	Group	Repeater	
<No.>	Control	Field	"No."
<Name>	Control	Field	Name
<Responsibility Center>	Control	Field	"Responsibility Center"
<Location Code>	Control	Field	"Location Code"
<VIP Code>	Control	Field	"VIP Code"
<Post Code>	Control	Field	
<Country/Region Code>	Control	Field	
<Phone No.>	Control	Field	
<Fax No.>	Control	Field	

Property	Value
BlankNumbers	<DontBlank>
BlankZero	<No>
AutoFormatType	<0>
AutoFormatExpr	<>
Emphasized	<FALSE>
SourceExpr	"VIP Code"
SubFormView	<Undefined>
SubFormLink	<Undefined>
ProviderID	<0>
TableRelation	"VIP Code"
Importance	<Typical>
CaptionExpression	<>
SkipOnTab	<No>
DrillDownFormID	<Undefined>
LookupFormID	<Undefined>
Lookup	<Undefined>
DrillDown	<Undefined>
AssistEdit	<Undefined>

2. In Page Designer, add a new line, **VIP Code**, to the **Customer List** group. This adds a new column to the **Customer List** page. Place the **VIP Code** line relative to where you want the column to be shown in the list.
3. To define the properties for the new element that you have added to the page, open the **Properties** window for the **VIP Code** line.
4. Set **SourceExpr** to **VIP Code**. The **VIP Code** field is in the **Customer** table, which is the source table for the **Customer List** place.
5. Save and compile the modified page. When you compile the page, the system is updated, and a RoleTailored client that connects to the database reads the updated page object.
6. Open the RoleTailored client and open the **Customer List** place to see the new **VIP Code** column.



	Responsibility ...	Location Code	VIP Code
eyer's Furnishi...		YELLOW	
sive Home Fur...		YELLOW	
ncepts Furnitu...		YELLOW	
y Canada Inc.		YELLOW	
Airport		YELLOW	
Candoxy Stor...		YELLOW	
nnon Group PLC	BIRMINGHAM	BLUE	
rian Ltd.			
rad Malaysia S...		YELLOW	
hts Electronics ...		GREEN	
ic Sdn Bhd		YELLOW	
s		YELLOW	
goce		BLUE	
MAROC		YELLOW	
Corp.		YELLOW	
upermarkets		YELLOW	
dit Fruit Export...		YELLOW	
addock Insuran...			
oulevard Kuite...		YELLOW	
n Meubelen		YELLOW	
y Nederland BV		YELLOW	
Zandpoort NV		YELLOW	
icopy		YELLOW	
Contractors		YELLOW	
natic		WHITE	

Customer Sales History ...

Customer No.:	01121212
Quotes:	0
Blanket Orders:	0
Orders:	0
Invoices:	0
Return Orders:	0
Credit Memos:	0
Pstd. Shipments:	0
Pstd. Invoices:	0
Pstd. Return Receip...	0
Pstd. Credit Memos:	0

Customer Statistics - Bil...

Customer No.:	01454545
Balance (LCY):	222.241,32
Outstanding Order...	702,82
Shipped Not Invd. (...	0,00
Outstanding Serv. ...	0,00
Shipped Not Invoic...	0,00
Outstanding Invoic...	0,00
Total (LCY):	222.944,14
Credit Limit (LCY):	0,00
Overdue Amounts (L...	
Balance Due (LCY):	0,00
Sales YTD (LCY):	0,00

Links

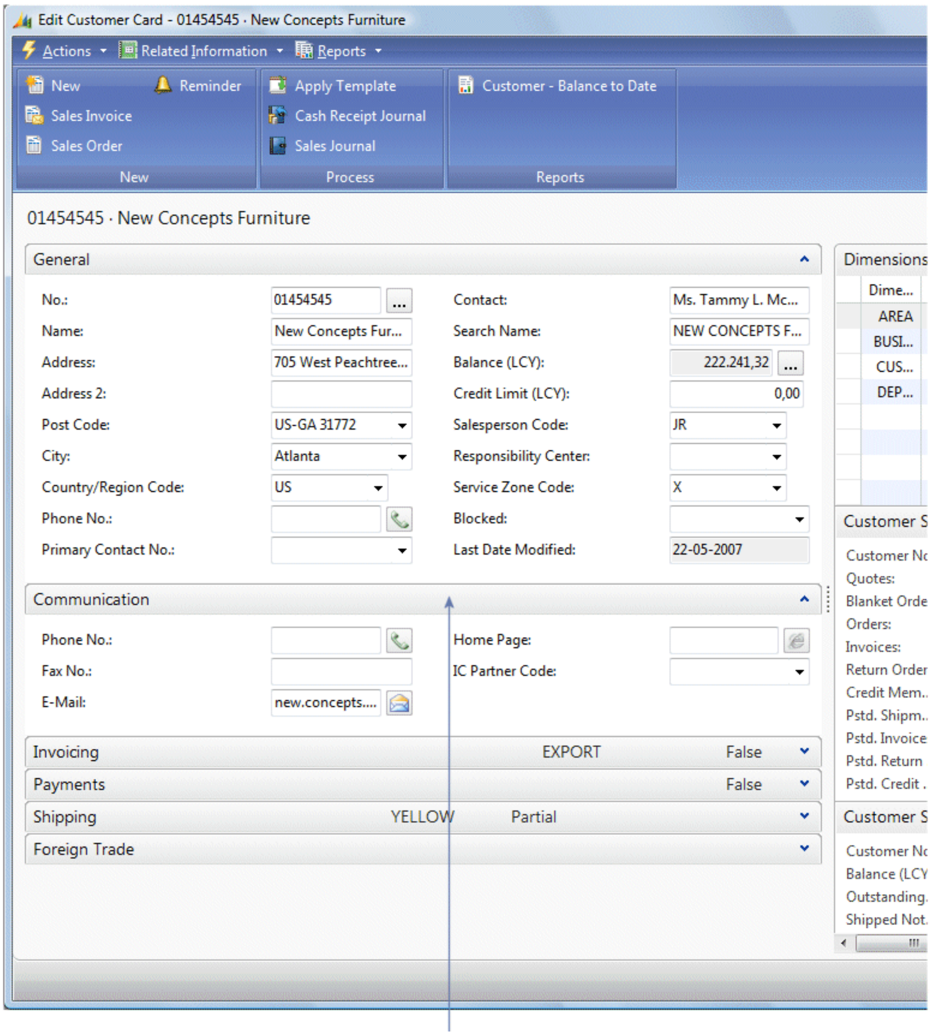
URL	7

To Modify the Customer Card Page

List places are read-only. To allow end users to view and update the **VIP Code** field, you must also modify the **Customer Card** page. Modifying the **Customer Card** page has some small differences, because a card is a different kind of page. Cards contain FastTabs and input fields instead of columns and rows.

1. In Object Designer, open the **Customer Card** page in Page Designer.
2. Place the cursor in the blank row at the bottom of Page Designer and open the **Properties** window to see which type of page this is.

   The **PageType** property determines the type of page and how the groups of elements on the page appear in the RoleTailored client. When the **PageType** property is **Card**, each group on the page is displayed as a FastTab. When the **PageType** property is **List**, as it was in the previous procedure, each group contains the columns that are displayed in a list.



01454545 · New Concepts Furniture

General

No.: 01454545
Name: New Concepts Fur...
Address: 705 West Peachtree...
Address 2:
Post Code: US-GA 31772
City: Atlanta
Country/Region Code: US
Phone No.:
Primary Contact No.:
Contact: Ms. Tammy L. Mc...
Search Name: NEW CONCEPTS F...
Balance (LCY): 222.241,32
Credit Limit (LCY): 0,00
Salesperson Code: JR
Responsibility Center:
Service Zone Code: X
Blocked:
Last Date Modified: 22-05-2007

Communication

Phone No.:
Fax No.:
E-Mail: new.concepts....
Home Page:
IC Partner Code:

Invoicing EXPORT False

Payments False

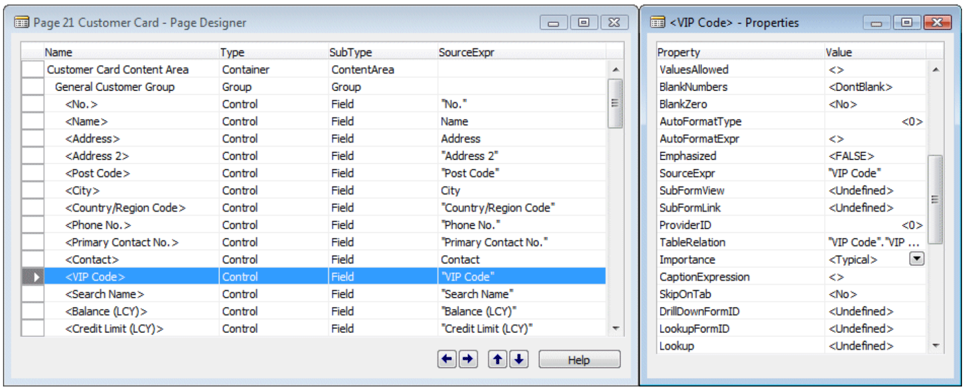
Shipping YELLOW Partial

Foreign Trade

FastTab

3. In Page Designer, add a new field to **General Customer Group** by inserting a row in the appropriate position.

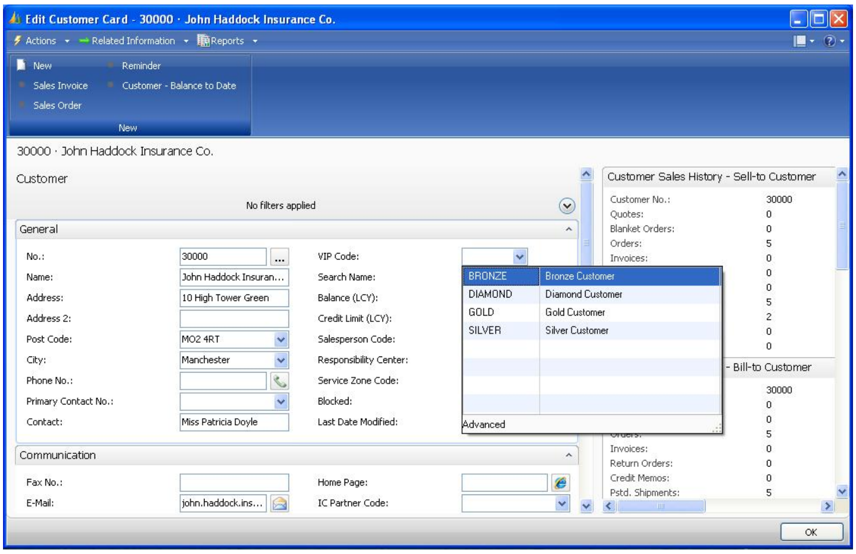
4. Open the **Properties** window and set the **VIP Code** field in the **Customer** table as the **SourceExpr** for the new **VIP Code** control in the **Customer Card** page. The **Customer** table is also the source table for the **Customer Card** page.



Name	Type	SubType	SourceExpr
Customer Card Content Area	Container	ContentArea	
General Customer Group	Group	Group	
<No.>	Control	Field	"No."
<Name>	Control	Field	Name
<Address>	Control	Field	Address
<Address 2>	Control	Field	"Address 2"
<Post Code>	Control	Field	"Post Code"
<City>	Control	Field	City
<Country/Region Code>	Control	Field	"Country/Region Code"
<Phone No.>	Control	Field	"Phone No."
<Primary Contact No.>	Control	Field	"Primary Contact No."
<Contact>	Control	Field	Contact
<VIP Code>	Control	Field	"VIP Code"
<Search Name>	Control	Field	"Search Name"
<Balance (LCY)>	Control	Field	"Balance (LCY)"
<Credit Limit (LCY)>	Control	Field	"Credit Limit (LCY)"

Property	Value
ValuesAllowed	<>
BlankNumbers	<DontBlank>
BlankZero	<No>
AutoFormatType	<0>
AutoFormatExpr	<>
Emphasized	<FALSE>
SourceExpr	"VIP Code"
SubFormView	<Undefined>
SubFormLink	<Undefined>
ProviderID	<0>
TableRelation	"VIP Code"."VIP ...
Importance	<Typical>
CaptionExpression	<>
SkipOnTab	<No>
DrillDownFormID	<Undefined>
LookupFormID	<Undefined>
Lookup	<Undefined>

5. Set the **TableRelation** property to “VIP Code” to enable the RoleTailored client to look up and display the choices that have been defined for this field.
6. Save and compile the modified page. When you compile the page, the system is updated, and a RoleTailored client that connects to the database reads the updated page object.
7. Open the RoleTailored client to see the modified **Customer Card** page.



30000 · John Haddock Insurance Co.

No.: 30000
Name: John Haddock Insuran...
Address: 10 High Tower Green
Address 2:
Post Code: MO2 4RT
City: Manchester
Phone No.:
Primary Contact No.:
Contact: Miss Patricia Doyle
VIP Code:
Search Name:
Balance (LCY):
Credit Limit (LCY):
Salesperson Code:
Responsibility Center:
Service Zone Code:
Blocked:
Last Date Modified:

BRONZE	Bronze Customer
DIAMOND	Diamond Customer
GOLD	Gold Customer
SILVER	Silver Customer

Fax No.:
E-Mail: john.haddock.ins...
Home Page:
IC Partner Code:

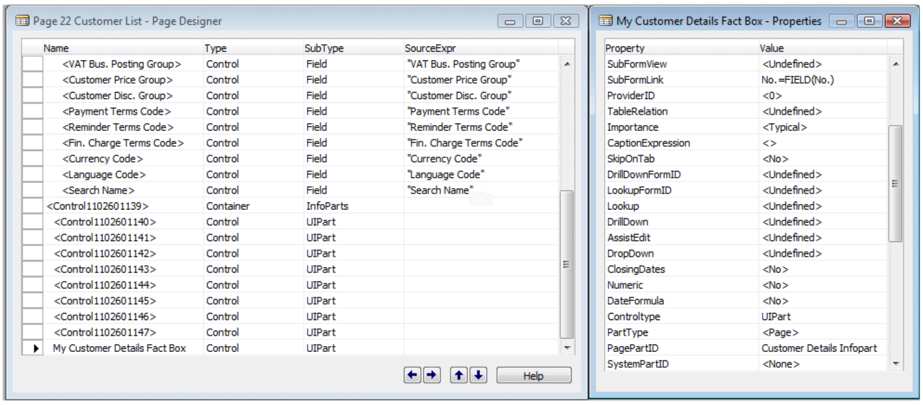
Customer No.: 30000
Quotes: 0
Blanket Orders: 0
Orders: 5
Invoices: 0

Invoices: 0
Return Orders: 0
Credit Memos: 0
Pstd. Shipments: 5

To Add a New FactBox to the Customer List Page

To complete this walkthrough, you create a new FactBox and add it to the **Customer List** page. The FactBox shows details about the customer that is currently selected in the list. By making this information readily available, the user has access to more information when they need it so that they do not need to open the **Customer Card** page to see basic information, such as phone number or e-mail address.

1. In Object Designer, open the **Customer List** page.
2. Insert a new row and enter **My Customer Details FactBox**.
3. Open the **Properties** window for this new row and ensure that the *PagePartID* property value is set to page 9084, which is **Customer Details FactBox**. This FactBox is shipped with the RoleTailored client.
4. Use the **SubFormLink** property in the **My Customer Details Fact Box - Properties** window to define the relation between the page and the source of the FactBox. FactBoxes are bound to the main contents of a page in the same way that forms and subforms are linked together.



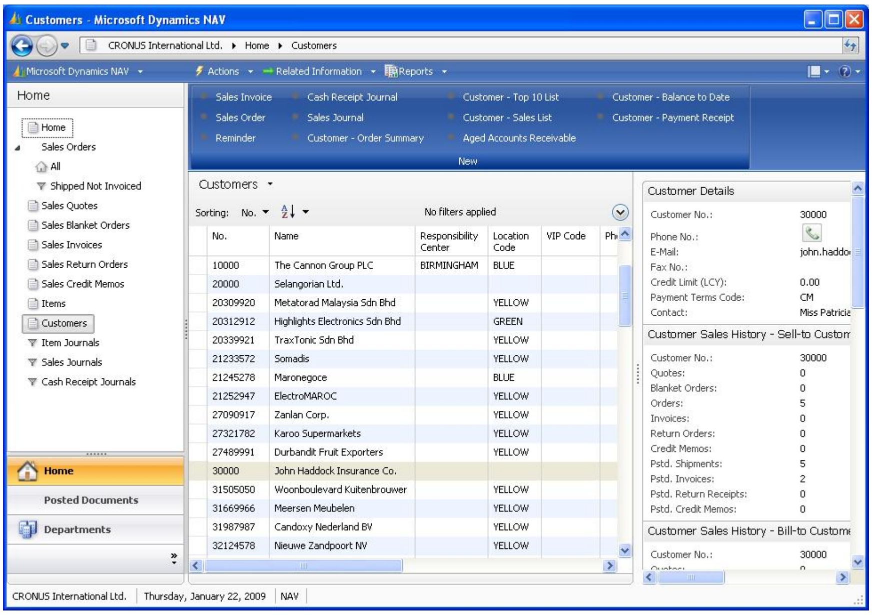
Page 22 Customer List - Page Designer
Help
My Customer Details Fact Box - Properties

Name	Type	SubType	SourceExpr
<VAT Bus. Posting Group>	Control	Field	"VAT Bus. Posting Group"
<Customer Price Group>	Control	Field	"Customer Price Group"
<Customer Disc. Group>	Control	Field	"Customer Disc. Group"
<Payment Terms Code>	Control	Field	"Payment Terms Code"
<Reminder Terms Code>	Control	Field	"Reminder Terms Code"
<Fin. Charge Terms Code>	Control	Field	"Fin. Charge Terms Code"
<Currency Code>	Control	Field	"Currency Code"
<Language Code>	Control	Field	"Language Code"
<Search Name>	Control	Field	"Search Name"
<Control1102601139>	Container	InfoParts	
<Control1102601140>	Control	UIPart	
<Control1102601141>	Control	UIPart	
<Control1102601142>	Control	UIPart	
<Control1102601143>	Control	UIPart	
<Control1102601144>	Control	UIPart	
<Control1102601145>	Control	UIPart	
<Control1102601146>	Control	UIPart	
<Control1102601147>	Control	UIPart	
My Customer Details Fact Box	Control	UIPart	

Property	Value
SubFormView	<Undefined>
SubFormLink	No.=FIELD(No.)
ProviderID	<0>
TableRelation	<Undefined>
Importance	<Typical>
CaptionExpression	<>
SkipOnTab	<No>
DrillDownFormID	<Undefined>
LookupFormID	<Undefined>
Lookup	<Undefined>
DrillDown	<Undefined>
AssistEdit	<Undefined>
DropDown	<Undefined>
ClosingDates	<No>
Numeric	<No>
DateFormula	<No>
Controltype	UIPart
PartType	<Page>
PagePartID	Customer Details Infopart
SystemPartID	<None>

5. Save and compile the modified page. When you compile the page, the system is updated, and a RoleTailored client that connects to the database reads the updated page object.

6. Open the RoleTailored client and open the **Customer List** page to see the changes.



No.	Name	Responsibility Center	Location Code	VIP Code	Ph
10000	The Cannon Group PLC	BIRMINGHAM	BLUE		
20000	Selangorian Ltd.				
20309920	Metatorad Malaysia Sdn Bhd		YELLOW		
20312912	Highlights Electronics Sdn Bhd		GREEN		
20339921	TraxTonic Sdn Bhd		YELLOW		
21233572	Somadis		YELLOW		
21245278	Maronegoce		BLUE		
21252947	ElectroMAROC		YELLOW		
27090917	Zanlan Corp.		YELLOW		
27321782	Karoo Supermarkets		YELLOW		
27489991	Durbandit Fruit Exporters		YELLOW		
30000	John Haddock Insurance Co.				
31505050	Woonboulevard Kuitenbrouwer		YELLOW		
31669966	Meersen Meubelen		YELLOW		
31987987	Candoxy Nederland BV		YELLOW		
32124578	Nieuwe Zandpoort NV		YELLOW		

Customer Details

Customer No.:	30000
Phone No.:	☎
E-Mail:	john.haddo
Fax No.:	
Credit Limit (LCY):	0.00
Payment Terms Code:	CM
Contact:	Miss Patricia

Customer Sales History - Sell-to Custom

Customer No.:	30000
Quotes:	0
Blanket Orders:	0
Orders:	5
Invoices:	0
Return Orders:	0
Credit Memos:	0
Pstd. Shipments:	5
Pstd. Invoices:	2
Pstd. Return Receipts:	0
Pstd. Credit Memos:	0

Customer Sales History - Bill-to Custome

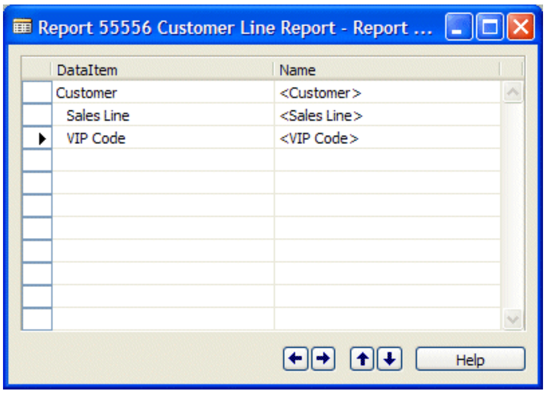
Customer No.:	30000

Step-by-Step Walkthrough: Creating a Report

In this walkthrough, you create a report that iterates through the **Customer** table. For each customer specified in the report filter, the report then iterates through the entire **Sales Line** table and lists these customers and the orders that they have placed.

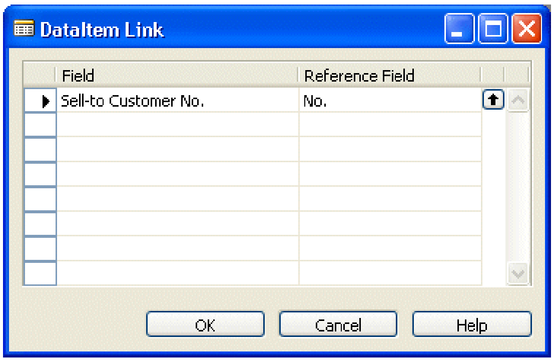
To Define the Data Model

1. In Report Designer, select the **Customer** table as the first data item, the **Sales Line** table as the second data item and the **VIP Code** table as the third data item. Indent the **Sales Line** and **VIP Code** data items:



DataItem	Name
Customer	<Customer>
Sales Line	<Sales Line>
VIP Code	<VIP Code>

2. Open the **Properties** window for the **Sales Line** data item. Set the **DataItemLinkReference** property to the name of the parent data item (**Customer**) to which the indented data item (**Sales Line**) must be related. In most cases, this is the default setting.
3. Set the **DataItemLink** property to point to the **Sell-to Customer No.** field from the **Sales Line** table. In the **Reference Field**, select the **No.** field from the **Customer** table.



Field	Reference Field
Sell-to Customer No.	No.

4. In the **Properties** window for the **VIP Code** data item set the **DataItemLink** property to point to the **VIP Code** field. In the **Reference Field**, select the **No.** field from the **Customer** table.
5. In the **Properties** window for the **Customer** data item, set the **PrintOnlyIfDetail** property to **Yes**. This ensures that the **Customer** body sections are only printed if there is data to print from the **Sales Line** table.

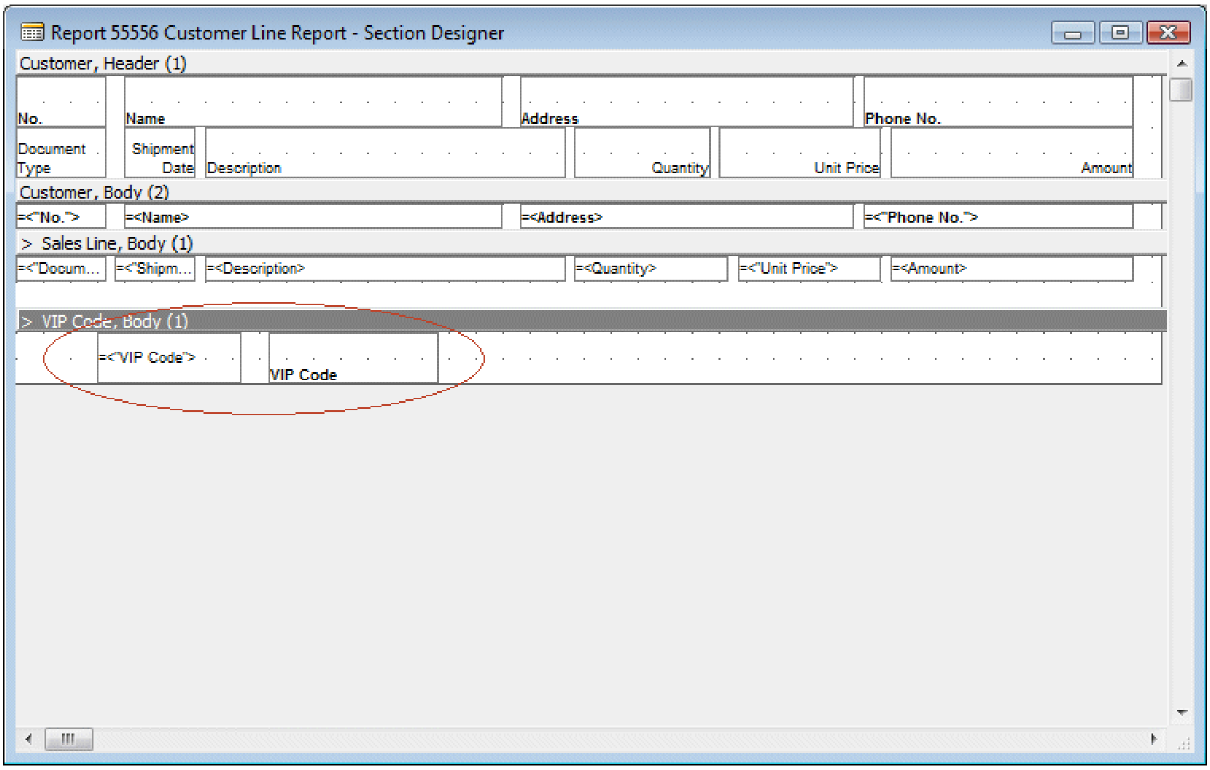
   The data model then works in the following way:

   - The report runs through the **Customer** data item.

- For each record in the **Customer** data item, records in the **Sales Line** data item are selected if the **Sell-to Customer No.** field has the same value as the **No.** field in the **Customer** data item.
- If there are no **Sales Line** records for a **Customer** data item, nothing is printed — not even the information from the **Customer** data item.
- It runs through the **VIP Code** table and prints the VIP code that has been assigned to each customer.

To Select the Fields that You Want to Appear in the Report

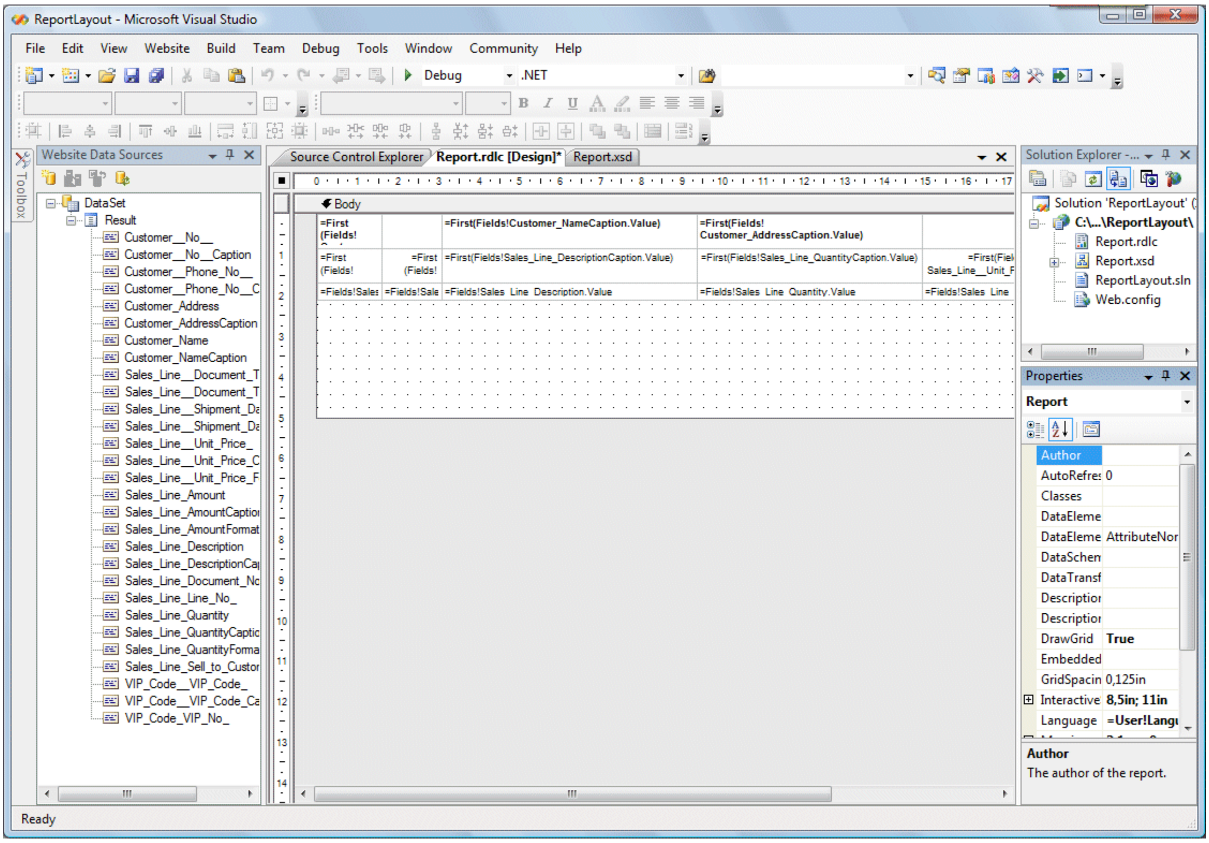
1. In Report Designer, select the **Customer** data item.
2. Open Section Designer, and then open the **Field Menu** window and add the **No.**, **Name**, **Address** and **Phone No.** fields.
3. Select the **Sales Line** data item and add the **Document No.**, **Shipment Date**, **Description**, **Quantity**, **Unit Price**, and **Amount** fields to the report.
4. Select the **VIP Code** data item and add the **VIP Code** field to the report. This is done in Section Designer for the report.



If you create sections in Report Designer, all fields that you add to these sections are displayed in the new report layout. Furthermore, all the fields included in the primary key of the tables that are used as data items in the report are displayed in the new report layout.

To Design the Report in Visual Studio

- Click **Tools**, **Transform Layout**. The layout of the report is transformed and opened in Report Designer in Visual Studio.



Note The dataset shows all fields from Microsoft Dynamics NAV that you specified in the earlier report designer. You can now use Visual Studio to design the layout of the report.

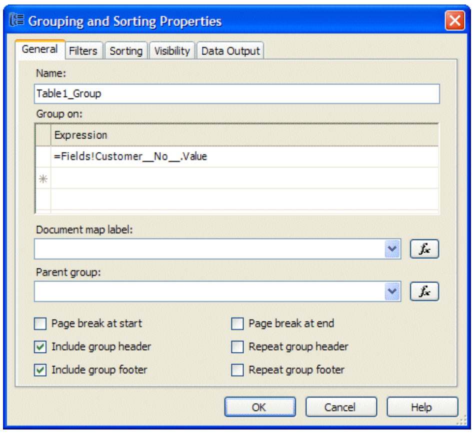
You design the layout of the report in Visual Studio and specify any grouping and totaling that you want in the report. It therefore makes sense to add any grouping and totaling to the report before you design the layout of the report.

To Add and Sort a Group and Sort the Detailed Entries

1. In Visual Studio, click any field in the report to enable the editors for the columns and rows.
2. Right-click a field of any row, click **Insert Group**, and then enter the following expression:

   ```
   =Fields!Customer_No_.Value
   ```

   This specifies that you want the entries to be grouped by customer number.



Name: Table1_Group

Group on:

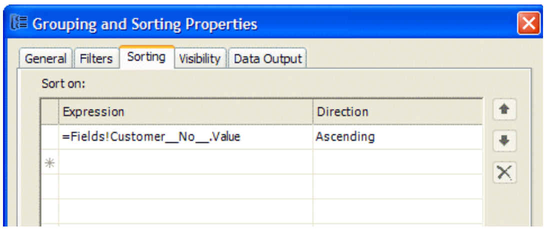
Expression
=Fields!Customer__No__.Value

☐ Page break at start ☐ Page break at end

☑ Include group header ☐ Repeat group header

☑ Include group footer ☐ Repeat group footer

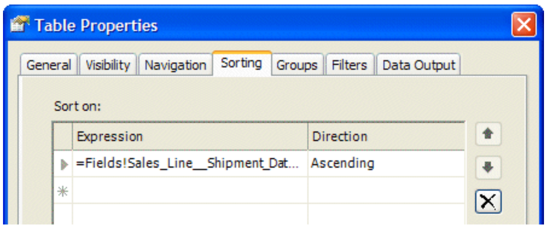
3. In the **Sorting** tab, select the field value by which you want the group sorted.



Sort on:

Expression	Direction
=Fields!Customer__No__.Value	Ascending

4. Open the **Table Properties** window.
5. In the **Sorting** tab, select the field value by which you want the detailed data entries sorted.



Sort on:

Expression	Direction
=Fields!Sales_Line__Shipment_Dat...	Ascending

You have now specified that you want the report to sort the sales line for each customer by their shipment date.

To Add a Subtotal

You can also add aggregate functions to your report in Visual Studio.

1. Add a row below the **Sales Line** row.
2. In the last cell in this new row, add the following expression:

   `=Sum(Fields!Sales_Line_Amount.Value)`

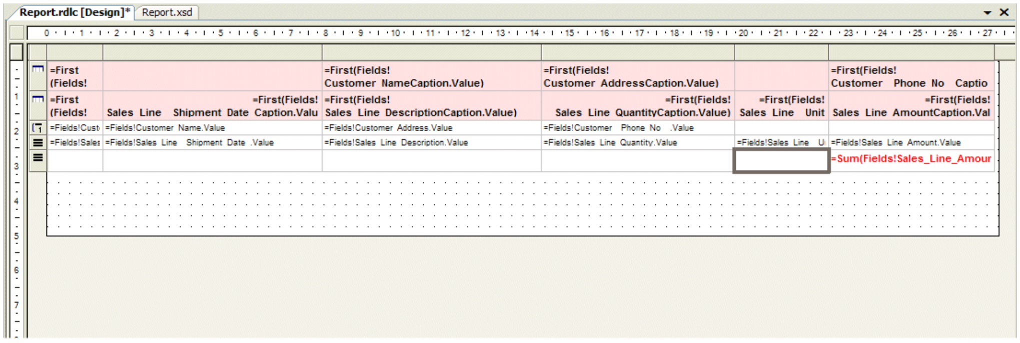
   This calculates the total amount ordered by each customer.

To Design the Layout of Your Report

You can easily format the appearance of many fields in your reports, including date and currency fields.

1. In the **Properties** window, select the **Format** tab.
2. Click the **More** (…) button. In the **Choose Format** window, select the format that you want the dates to appear in.
3. To format the table headers so that they stand out from the rest of the report, select the header rows.

You can now change the style, color, size, and so on of the font as well as the background color of the cells. When you are finished the report should look something like this:



=First (Fields!		=First(Fields! Customer NameCaption.Value)	=First(Fields! Customer AddressCaption.Value)		=First(Fields! Customer Phone No Captio
=First (Fields!	=First(Fields! Sales Line Shipment Date Caption.Valu	=First(Fields! Sales Line DescriptionCaption.Value)	=First(Fields! Sales Line QuantityCaption.Value)	=First(Fields! Sales Line Unit	=First(Fields! Sales Line AmountCaption.Val
=Fields!Cust	=Fields!Customer Name.Value	=Fields!Customer Address.Value	=Fields!Customer Phone No .Value		
=Fields!Sales	=Fields!Sales Line Shipment Date .Value	=Fields!Sales Line Description.Value	=Fields!Sales Line Quantity.Value	=Fields!Sales Line U	=Fields!Sales Line Amount.Value
					=Sum(Fields!Sales_Line_Amour

4. When you are done with the report layout, save and close the project.
5. In the Classic client, when you open this report in Report Designer, a message appears informing you that the `.rdlc` file for this report has changed and asks you if you want to load the changes. Click **Yes** to save the changes in the database.
6. Compile the report.

The database now contains two versions of this report:

- C/SIDE version that can be displayed in the Classic client.
- SQL Server Reporting Services version that can be displayed in the RoleTailored client.

Important If you want to further modify the design of the report, you must open the report in the C/SIDE Report Designer and click **View**, **Layout**. If you click **Tools**, **Transform Layout**, then you will convert the original C/SIDE version of the report again and will not be able to see the changes that you have already made.

Also, the C/SIDE Report Designer and Visual Studio are not closely integrated. Any changes made in one environment will not be visible in the other environment if performed independently. You must save any changes that you make in either development environment and import or export the changes in the other environment before they will be visible there.

Appendix

Forms

The following properties and functions are not supported in pages and are removed when using the Form Transformation Tool to generate a page from a form. All code and properties that are removed are recorded in the Transaction.log output file.

Form Properties

To be added when finalized

Report Functions

To be added when finalized

Reports

The following properties and functions are not supported in reports for the RoleTailored client:

Report Properties

- BottomMargin
- DeviceFontName
- HorzGrid
- LeftMargin
- Orientation
- PaperSize
- PaperSourceFirstPage
- PaperSourceOtherPages
- RightMargin
- ShowPrintStatus
- TopMargin
- VertGrid

Data Item Properties

- GroupTotalFields
- NewPagePerGroup
- NewPagePerRecord
- TotalFields

Report Functions

- NewPagePerRecord
- ObjectID
- URL

CurrReport Functions

- CreateTotals
- NewPage
- NewPagePerRecord
- ObjectID
- PageNo
- PaperSource
- SaveAsXML
- ShowOutput
- TotalsCausedBy
- URL

Report Triggers/Functions

- Section triggers
- Layout-specific functions and totaling functions
- Unsupported functions referred to from other objects (codeunits, tables, and so on)

XMLports and Dataports

The following properties are added to XMLports to enable dataport functionality for the RoleTailored client.

New XMLport Properties

- FieldDelimiter
- FieldSeparator
- Format
- RecordSeparator
- TableSeparator
- UseRequestForm
- Width

XMLport Functions Not Carried Over From Dataports

- OnBeforeExportRecord

Microsoft Dynamics is a line of integrated, adaptable business management solutions that enables you and your people to make business decisions with greater confidence. Microsoft Dynamics works like and with familiar Microsoft software, automating and streamlining financial, customer relationship and supply chain processes in a way that helps you drive business success.

U.S. and Canada Toll Free 1-888-477-7989
Worldwide +1-701-281-6500
www.microsoft.com/dynamics