

Development

Microsoft Dynamics™ NAV

The Development Environment in Microsoft Dynamics™ NAV

Date: Feb, 2007



Table of Contents

The Development Environment in Microsoft Dynamics™ NAV.....	1
Introduction	3
The Three-Tier Architecture	3
The New UI	4
Designing for the New User Interface	5
Using the Development Environment	7
Page Designer	8
Personalization	16
Report Designer.....	17
Creating a Report	17
Web Services	22
Working with Microsoft Dynamics NAV Web Services	23
Service-enabling Codeunits that Expose Complex Data	24
Migrating to the Microsoft Dynamics NAV Role Tailored Client.....	25
The Transformation Tool	25
Methodologies	26
Features that have been Redesigned	28
Appendix	34

Introduction

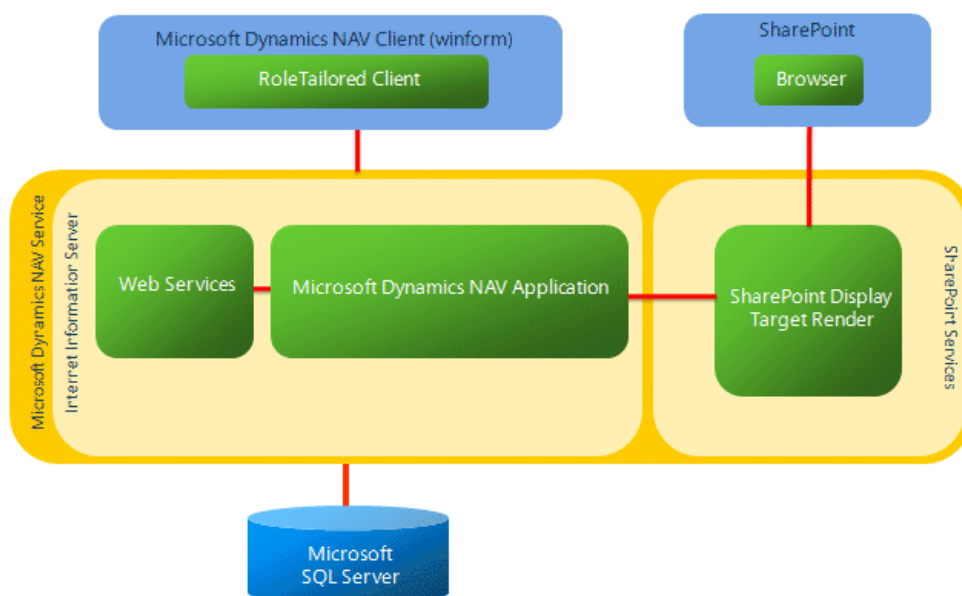
The next release of Microsoft Dynamics NAV introduces some significant changes in the underlying architecture. The Dynamics NAV Service Architecture adds a true middle tier server that runs exclusively on Microsoft SQL Server and uses Web services as the communication protocol for Microsoft Dynamics NAV clients that connect with it. This document describes the underlying architecture, the new features in the development environment and explains the main benefits that are gained by writing applications for the Service Architecture. Despite the extensive changes that have been made to the system infrastructure, the core aspects of developing solutions for Microsoft Dynamics NAV remain unchanged. Indeed the Object Designer has been extended to incorporate new features that allow you to create objects for the new User Interface, Web Services and powerful new reports.

The Three-Tier Architecture

The Dynamics NAV Service Architecture is a classic 3 tier model with Microsoft SQL Server providing the data layer, the Microsoft Dynamics NAV Service providing the business logic layer and the new thin client providing the front end.

The Service runs as a Web service enabled application on Microsoft Internet Information Services. As a Web service system, the Service can serve multiple clients and support different types of client. This release supports a Winforms client (known as the RoleTailored Client) and a SharePoint client (the Microsoft Dynamics™ NAV Portal for Microsoft SharePoint®) as well as inherently supporting any custom built Web Services that are made and published by developers for this system.

The fundamental difference between the existing C/SIDE architecture and the new Service architecture is that the business logic now runs on the Service and no longer runs on the client. In the C/SIDE architecture, the client is responsible for both the presentation layer (Forms) and executing the C/AL code. In the new Service Architecture, clients are responsible for rendering the presentation (Pages) and for forwarding requests to the Service to run C/AL code. The Service is capable of executing multiple client requests in parallel as well as serving other clients by providing Web service access to authenticated clients that are running the RoleTailored Client, the Microsoft Dynamics NAV Portal and Web Service clients.



The Service also acts as an additional layer of security between the clients and the database. It leverages the authentication features of Microsoft Internet Information Services (IIS) as another layer of user authentication while using impersonation to ensure that the business logic is executed in a process that has been instantiated by the user who submitted the request. This ensures that the direct and indirect permissions that have been granted to the user in the Microsoft Dynamics NAV security system are respected, as well as maintaining an audit trail of user activity.

There are two significant changes to the underlying system that are worth mentioning even though they are both hidden from developers and users. Firstly, while you still write all your code in the Object Designer, the C/AL code is no longer executed within the Dynamics NAV client runtime. Instead, C/AL is converted at compilation time to a .NET language before being compiled into a Microsoft Intermediate Language (MSIL) assembly. This assembly is linked with private libraries that continue to provide the functions offered by the Dynamics NAV runtime. The impact of this change is negligible for developers – the assemblies can be compiled independently because they are late bound and the code runs with the same result as always. Microsoft Dynamics NAV has taken an enormous step forward by supporting .NET and in the future we will expose more native .NET functionality to developers and possibly even allow C/AL functions to be accessed from other .NET applications!

The second important change is a result of moving to more model driven development that relies heavily on Metadata. In the Classic Client, forms are designed in a WYSIWYG fashion. The RoleTailored Client displays information in Pages and creating a Page is more like writing an HTML page. The Page elements are listed in their relative display order and the properties associated with each element are used to specify special presentation features, thereby allowing the client to decide how to best display the page. This small change in focus from WYSIWYG to relative grouping actually allows a page to be consumed by different clients. It's up to each particular client to determine how to render the page. Metadata is not only restricted to Pages. Function descriptions in Codeunits, Table definitions and any type of properties on objects are all examples of Metadata. Understanding that clients speak to the server in terms of metadata requests helps explain some of the system's behavior. For example, writing Pages feels more like writing XML and the new field properties in tables change the way the RoleTailored Client displays data.

The new architecture provides greater scalability by allowing you to install several Services that access the same database and also allows you to balance your system by dedicating different Services to particular application areas or to different clients.

The New UI

One of the key features of the new Microsoft Dynamics NAV release is the RoleTailored Client. After extensive research into understanding how our customers work, their business goals and their departmental organization, Microsoft Dynamics has created a new interface with new behavior that supports working in a role oriented, task focused manner.

We have introduced a new object type, the Page, which you design in the Object Designer and can be displayed on clients that run on the Microsoft Dynamics Service. Pages contain new types of controls that enable the advanced representation of system data and shortcuts to system features.

There are no specific subtypes of Pages but just as you are able to build card and list forms today, there are some broad generalizations in the type of Page that you can make for the RoleTailored Client. These categories spring from a User Experience research program that has been carried out within Microsoft Dynamics and aims to provide a unified UI to the users of all the Dynamics applications.

The general types of Page include:

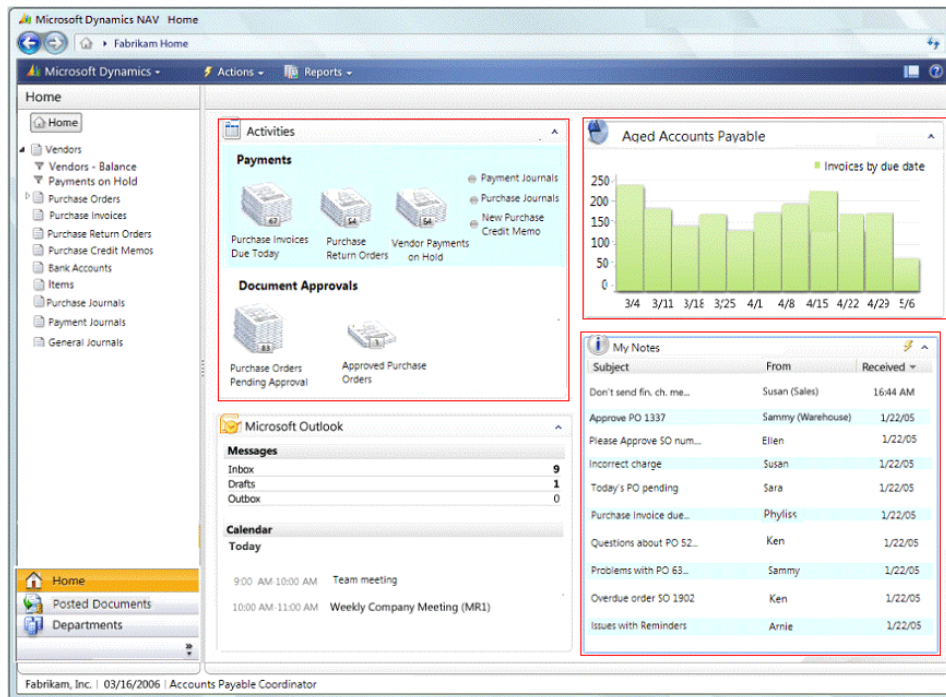
- **Role Center** – This type of page is the starting point for the user. It links the user to all the tasks and processes that constitute their role in the company. The role center should provide a task-oriented navigation structure that reduces the complexity of the information that the user has to sift-through and allows them to focus on the key work areas that are associated with their role. You should build each individual Role Center with a specific type of user in mind.
- **List places** – This type of page helps users focus and prioritize their work by showing the tasks they must perform and the related information that they need to perform these tasks together in the same window. List places are very similar to List Forms in the Classic Client.
- **Task pages** – This type of page contains all the tasks that are part of each process. For example, the Quick Access 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.
- **Fact boxes** – This type of page is like a mini-card that provides supplemental information to List places, Task pages and Role Centers. Fact Boxes reduce the amount of time that the user spends searching by presenting all the information that they need for a given context in one window.

Designing for the New User Interface

Many of the concepts involved in creating Pages are similar to those involved in creating Forms for the Classic Client. When you design for the RoleTailored Client, you must consider which users are most likely to use the page and what information they will need to access. The source table for the Page should be the table that contains the primary set of data that the Page will display.

Additionally, you should consider any supplemental sources of information that will add value to the Page and add these as Fact Boxes or add any existing Fact Boxes that support your design. Lastly, you should add links to the Navigation Bar so that users can access this new Page.

The following Role Center highlights three Fact Boxes each of which draws in different information for the user. The stacks of paper seen in the central Fact Box are known as Cues. When you click a cue, it opens a task page where you can perform the function associated with that cue. Completing tasks from Cues reduces the size of the paper stack and allows users to see how much work they have left to do.



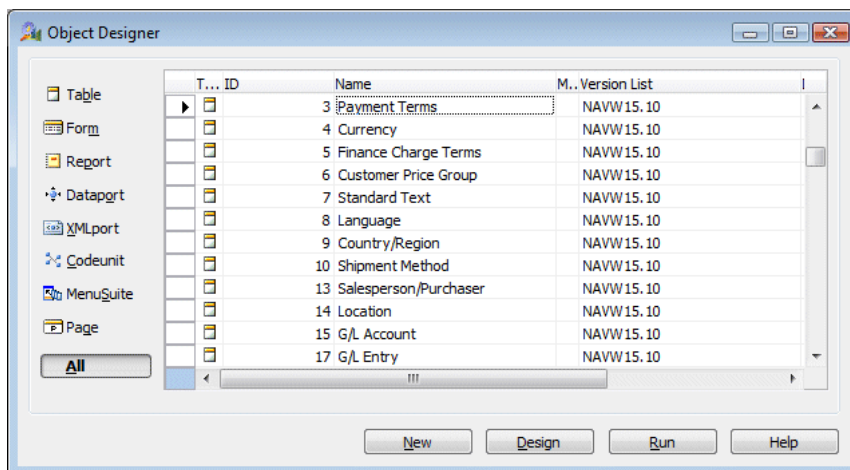
The Accounts Payable Coordinator's Role Center

Notice also how Microsoft Outlook is integrated into the Role Centre. Adding a user's Outlook Fact Box to their Role Centre is as easy as creating a new Fact Box for them. These are just a few of the new features that are available when you build applications for Microsoft Dynamics NAV.

Using the Development Environment

You have always used the Object Designer to develop solutions for Microsoft Dynamics NAV and the new development environment features are also available in the Object Designer.

The following screenshot shows the new designers and the subsequent table summarizes the changes that have been made to the designer and compares the runtime behavior of the objects in the Classic Client and their behavior in the RoleTailored Client.

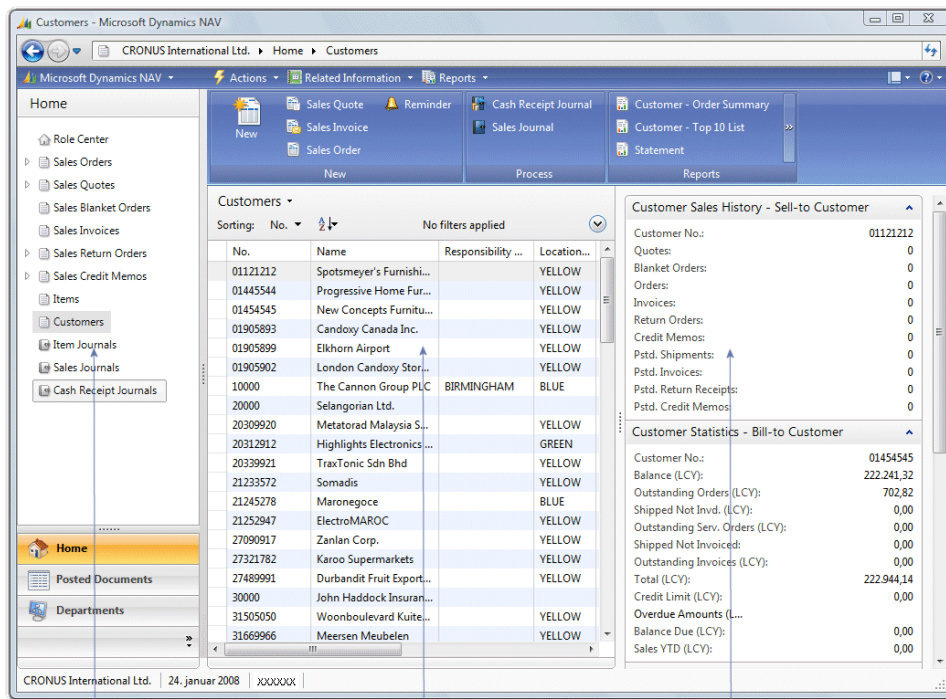


Object	Service Architecture	Classic Client
Table	Tables are designed and executed in the same way as they always have been in the Classic Client. All the same triggers and properties are available. Some new properties such as ExtendedDataType allow you to add metadata to fields and this changes how the fields are rendered on the RoleTailored Client.	Same as previous versions of the Classic Client. Some new properties such as ExtendedDataType have been added but have no affect on the behavior of tables in the Classic Client.
Forms	Not supported.	Same as previous versions of the Classic Client. All code on Forms continues to run.
Pages	New object type that contains many properties and methods like those used by forms. The Page Designer looks similar to the XMLport Designer.	Not supported.
Web Services	Web Services are designed by creating or reusing a Page or Page and Codeunit objects together. XMLports are used to parameterize complex data types through Codeunits .	Not supported.
Codeunits	Codeunits are designed and executed in the same way as they always have been in the Classic Client. The key	Same as previous versions of the Classic Client.

	<p>difference is that code now runs on the Service and not on the client. Any functions that run on the local computer use the Service environment rather than the RoleTailored Client.</p> <p>New functions have been added to stream files and data between the RoleTailored Client and the Service.</p>	<p>New functions have been added to stream files and data between the RoleTailored Client and the Service. These functions can run on the Classic Client but do nothing.</p>
Reports	<p>Report datasets and sections are designed in the C/SIDE Report Designer.</p> <p>The report layout is designed using a Report Definition Language (RDL) editor like Visual Studio. The layout information from the RDL file is stored in the new properties in the Report Object.</p>	<p>Same as previous versions of the Classic Client.</p> <p>Reports have 3 new properties that are ignored on the C/SIDE client but contain report layout information for reports that run on the Service Architecture.</p>
Dataports	Not supported.	Same as previous versions of the Classic Client.
XMLports	<p>XMLports are designed and execute as they do in the Classic Client. Furthermore, they have been extended with functionality similar to Dataports to cover the ability to import and export structured files.</p>	Same as previous versions of the Classic Client.
MenuSuite	MenuSuite behaves in the same way in both the RoleTailored Client and in the Classic Client.	Same as previous versions of the Classic Client.

Page Designer

The Microsoft Dynamics NAV RoleTailored Client displays data in pages. There are several different kinds of pages, including, Role Centers, List Pages and Fact Boxes. These pages are designed in the Page Designer on the Classic Client.



Navigation Pane

List Page

Fact Boxes

Once you have decided what the design of a page should be; creating it is simple using the 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 of the elements in the page.

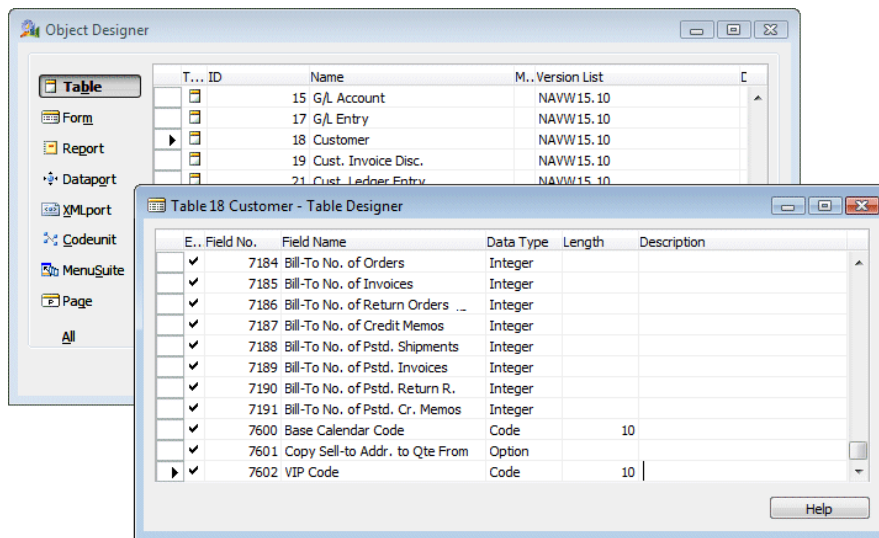
By only specifying the content that should be shown and how it should be presented, you are freed from the static layout of form-based designers and the tedious pixel-level editing that they entail. The central activity in page design is specifying the contents of the page in relation to the different Microsoft Dynamics NAV page types: List, Card, Journal, and Worksheet.

In the following example, you will see how to create and modify pages in Microsoft Dynamics NAV. The example starts by adding a new field to the **Customer** table and goes on to show the changes that must be made to the **Customer List** page and the **Customer Card** page before they can display this new table field. Lastly, you are shown how to create a new Fact Box that displays some details about the customer that is currently selected in the **Customer List** page.

Step 1: Adding a New Field to the Customer Table

For the following examples, we have created a new table called VIP. This table contains the VIP status settings (*Bronze, Silver, Gold, and Diamond*) that you can assign to your customers. In the following examples, we will show you how to integrate this table into the Microsoft Dynamics NAV application.

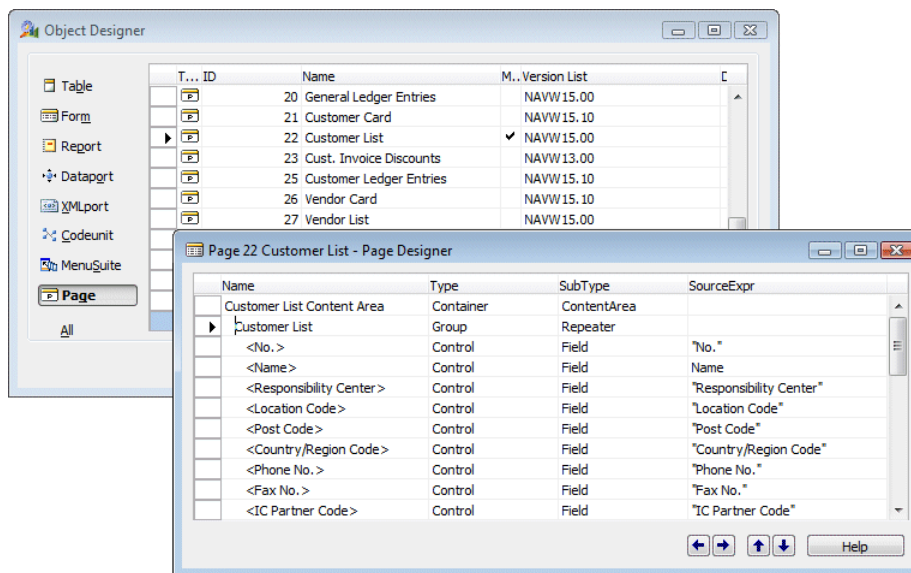
Open a Classic Client and use the Table Designer to extend the **Customer** table by adding a field called **VIP Code**. This field contains information about each customer's VIP status. Make the **VIP Code** field link to the **VIP Code** table.



Step 2: Modifying the Customer List Page

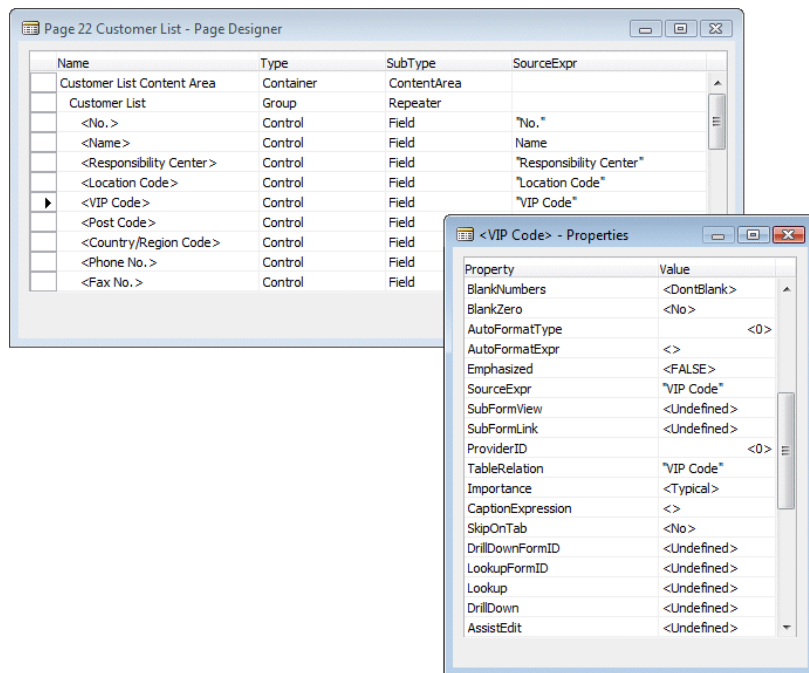
The next step is to add the information about the new **VIP Code** field to the **Customer List** page.

In the Object Designer, Open the **Customer List** page in the new Page Designer.



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 the Page Designer defines the page's primary group and in this example it is called *Customer List Content Area*. This element has the subtype *ContentArea*. This means that the contents of this group are displayed in the Content Area when it is displayed in the Microsoft Dynamics NAV role-based client. Other important subtypes are *InfoParts* and *HomeParts*. These subtypes are used later in this example.

The hierarchical structure of the page is determined by the indentation of the rows shown in the Page Designer. *Customer List* has elements 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.

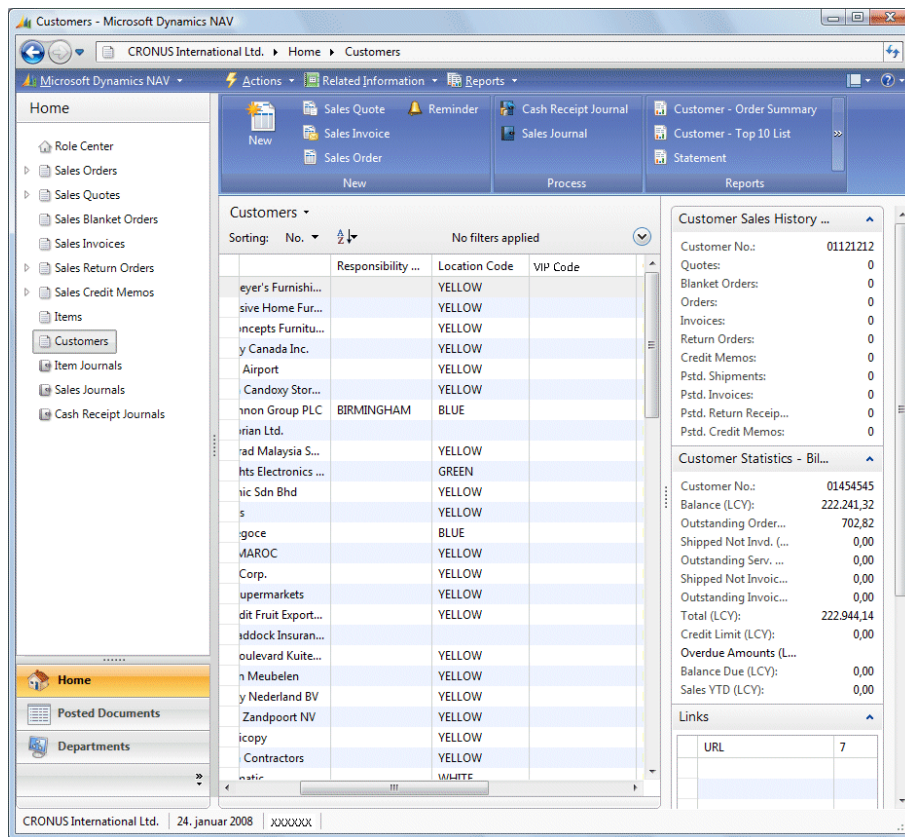


In the Page Designer, add a new line, *VIP Code* to the *Customer List* group. This adds a new column to the **Customer List** page. Place the *VIP Code* relative to where you would like the column shown in the list.

To define the properties for the new element you have added to the page, open the **Properties** window for *VIP Code*. The **Customer** table is the source table for the **Customer List** page and you must set the **VIP Code** field in the **Customer** table as the *SourceExpr* for the new *VIP Code* control.

After you have added the **VIP Code** field to the *Customer List* group, you must compile the modified page. When you compile the page, the system is updated and a RoleTailored Client that connects to the database will read the updated Page object.

To see the changes, open the RoleTailored Client, and open the **Customer List** and note the new VIP Code column.

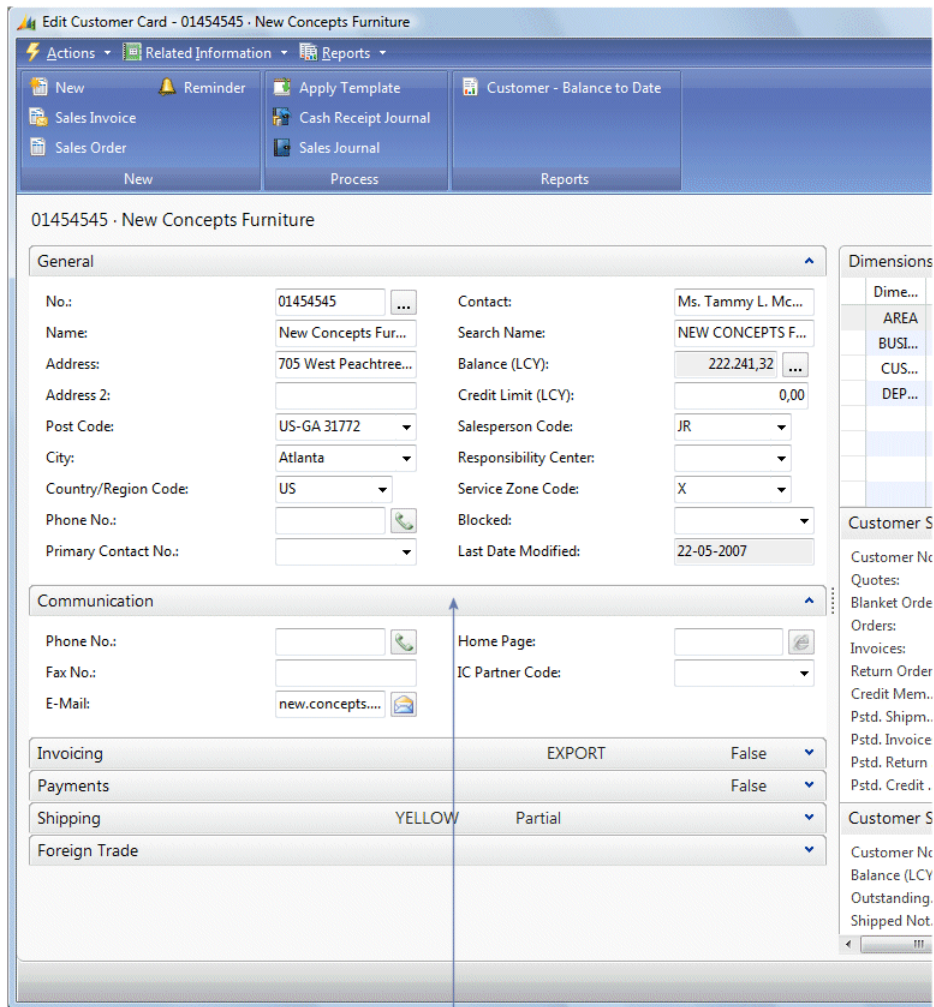


Step 3: Modifying the Customer Card Page

List places are non editable. 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 is similar to the previous steps but there are some minor differences, mainly because a card is a different kind of page. Cards contain Fast Tabs and input fields rather than the columns and rows that lists have.

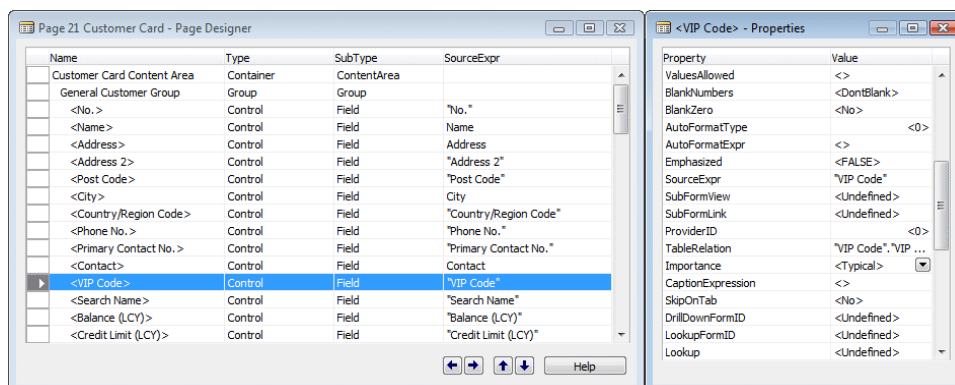
In the Object Designer, open the **Customer Card** in the Page Designer. Place the cursor in the blank row at the bottom of the Page Designer and open the **Properties** window to see which type of page this is. It is the **PageType** property that determines which type of page this is and how the groups of elements in the page appear in the RoleTailored Client.

For example, when the **PageType** is Card, each group in the page is displayed as a *Fast Tab* and when the **PageType** is List, as it was in the previous example, each group contains the columns that are displayed in a list.



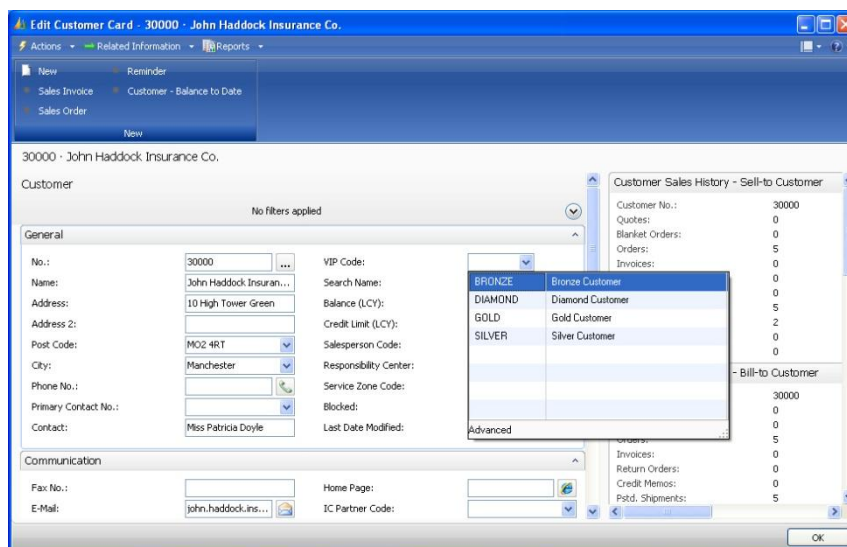
Fast Tabs

In the Page Designer, add a new field to *General Customer Group* by inserting a row in the appropriate position. Open the **Properties** window and modify the properties of this new field. The **Customer** table is also the source table for the **Customer Card** page. Set the **VIP Code** field in the **Customer** table as the *SourceExpr* for the new VIP Code control in the **Customer Card**.



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.

Save and compile the page object and the modified **Customer Card** is automatically deployed on the service tier.

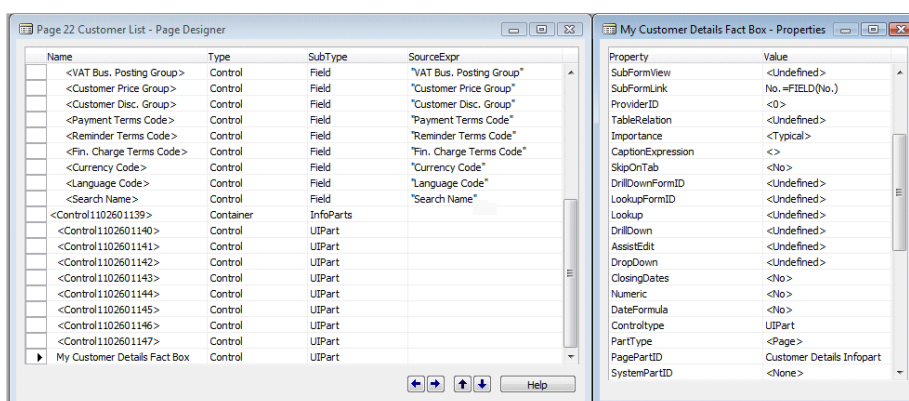


Step 4: Adding a New Fact Box to the Customer List Page

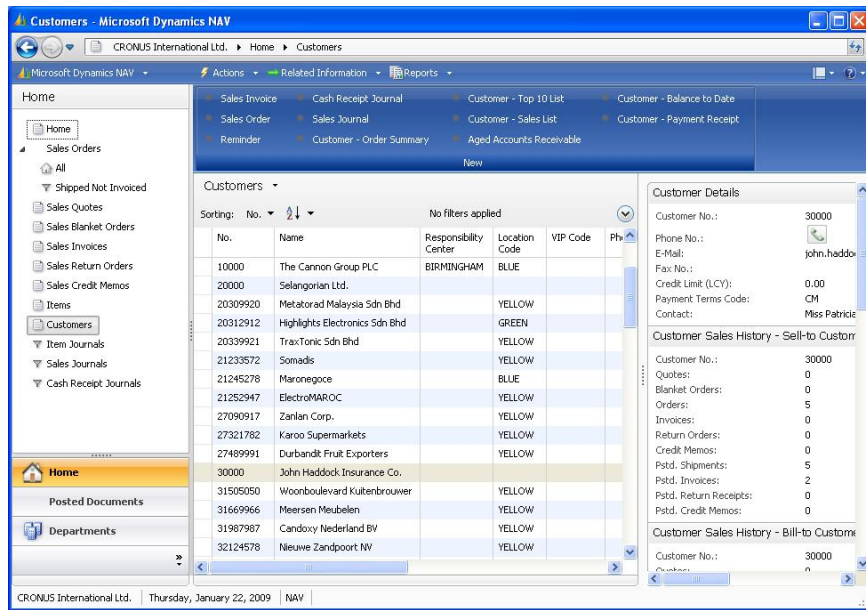
To complete this example, we must create a new Fact Box and add it to the **Customer List** page. The Fact Box should show details about the customer that is currently selected in the list. By making this information readily available, we ensure that the end-user has access more information when they need it and save them from having to open the **Customer Card** just to see some basic information such as phone number, e-mail address, and so on.

In the Object Designer, open the **Customer List** Insert a new row and call it *My Customer Details Fact Box*. Open the **Properties** window for this new row and ensure that the *PagePartID* property value is set to page 9084, **Customer Details InfoPart**. This is one of the Fact Boxes that are shipped with Microsoft Dynamics NAV.

Fact Boxes are bound to the main contents of a page in much the same way as forms and subforms are linked together. 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 Fact Box.



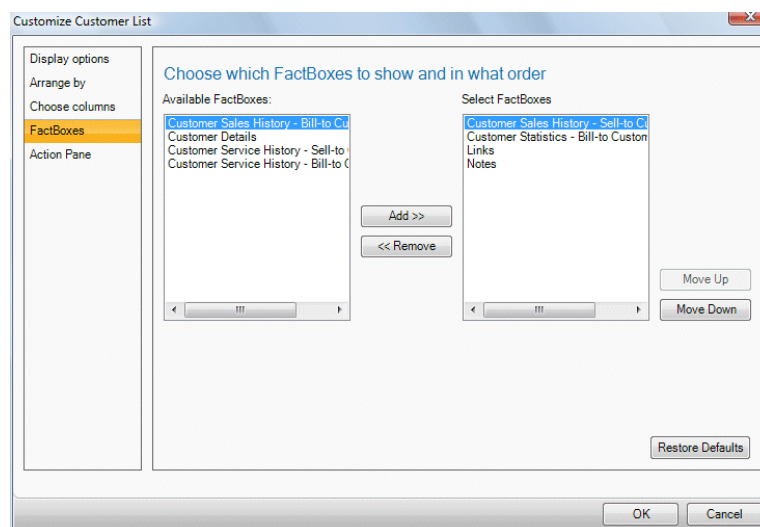
Close, save and compile the **Customer List** page and open the RoleTailored Client to see the changes that we have made.



Personalization

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 Fact Boxes should be displayed. Similarly end-users can change the display order of Fact Boxes – or choose to hide non-relevant Fact Boxes for their Role Center or a particular Page.

When you are designing pages in Microsoft Dynamics NAV you should keep the fact that users can customize the final page in mind. The Personalization tool does not allow users to access any parts of the system that they have not been given permission to. However, it does let users remove page elements that they would prefer not to see. This means that as a page designer, you should include a comprehensive list of page elements and keep in mind that the most common changes a user will make is to remove elements from rather than add elements to the page you're designing.



Report Designer

Designing reports for Microsoft Dynamics NAV clients is different depending on whether you are targeting the RoleTailored Client or building reports for the Classic Client. Although the two ways of designing reports are different, some steps are the same and all reports are stored in the same type of object. It is also possible to build a single report object that can run on both the Classic Client and on the RoleTailored Client.

The extra information that the Role Tailored Client needs is recorded in the Report Definition Language (RDL) and stored in some extra properties in the report object. Because of the way reports are generated for the two different client types, there are other changes that should be considered. The most significant of these is that the layout of the reports is defined differently. When you design a report for the Classic Client, you design the layout in the Section Designer. When you design a report for the RoleTailored Client, you design the layout in an RDL editor such as Visual Studio.

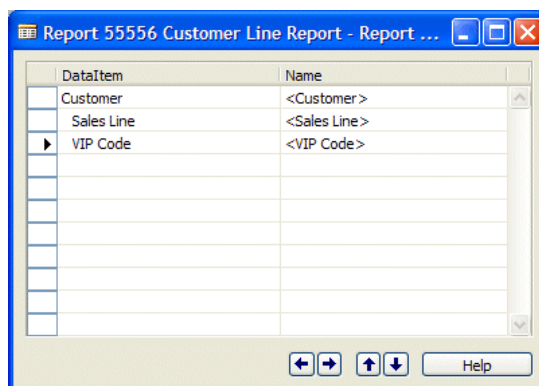
To create a report for the RoleTailored Client, you define the data items for the report in the usual way and then export the report to Visual Studio. Visual Studio is used to design the layout of the report, as well as any post processing such as grouping and totalling. These changes mean that there are some properties that are not supported in the new architecture. In general, these are the properties that deal with the layout of the report, totalling, grouping and printer specifications.

Creating a Report

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

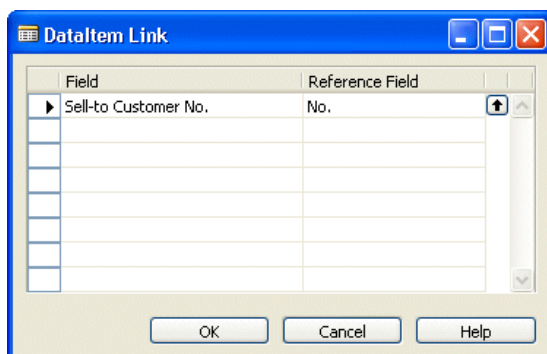
Step 1: Defining the Data Model

In the 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:



Open the **Properties** window for the **Sales Line** data item. Set the *DataItemLinkReference* property to the name of the parent data item (Customer) that the indented data item (Sales Line) must be related to. In most cases, this is the default.

Set the *DataltemLink* 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.



In the **Properties** window for the VIP Code data item set the *DataltemLink* property to point to the **VIP Code** field. In the **Reference Field**, select the **No.** field from the **Customer** table.

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 now works like this:

- 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, 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.

Now you can select the fields that you want to appear in the report.

In the Report Designer, select the **Customer** data item, open the Section Designer and then open the **Field Menu** window and add the **No.**, **Name**, **Address** and **Phone No.** fields.

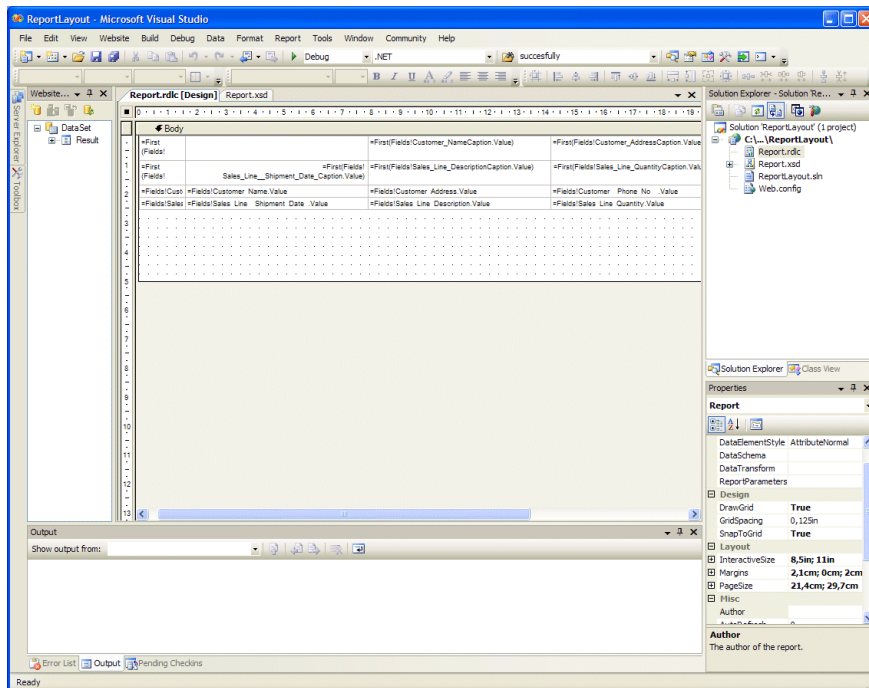
Select the **Sales Line** data item and add the **Document No.**, **Shipment Date**, **Description**, **Quantity**, **Unit Price** and **Amount** fields to the report.

Select the **VIP Code** data item and add the **VIP Code** field to the report.

Note

If you create sections in the Microsoft Dynamics NAV Report Designer, all the fields that you add to these sections will appear 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 will appear in the new report layout.

Click Tools, Transform Layout and the layout of the report is transformed and opened in Visual Studio Report Designer:



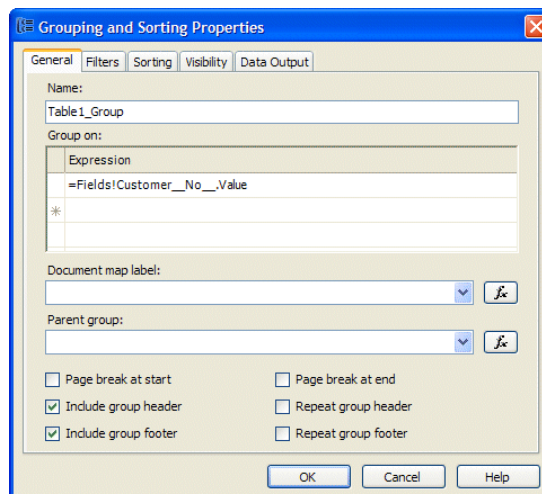
You can now design the layout of the report.

Step 2: Designing the Report

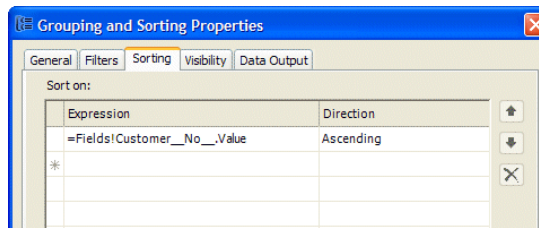
You design the layout of the report in Visual Studio as well as specify any grouping and totalling that you want the report to perform. It therefore makes sense to add any grouping and totalling to the report before you design the layout of the report.

Adding a Group

In Visual Studio, click any field in the report to make the handles for the columns and rows appear. Right-click the handle of any row, click Insert Group and enter the value shown in the following screenshot to specify that you want the entries grouped by customer number.

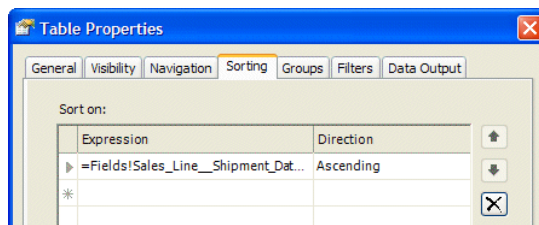


In the **Sorting** tab, select the field value that you want the group sorted by.



Sorting the Detailed Entries in the Report

Open the Table Properties window and in the **Sorting** tab, select the field value that you want the detailed data entries sorted by.



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

Adding a Subtotal

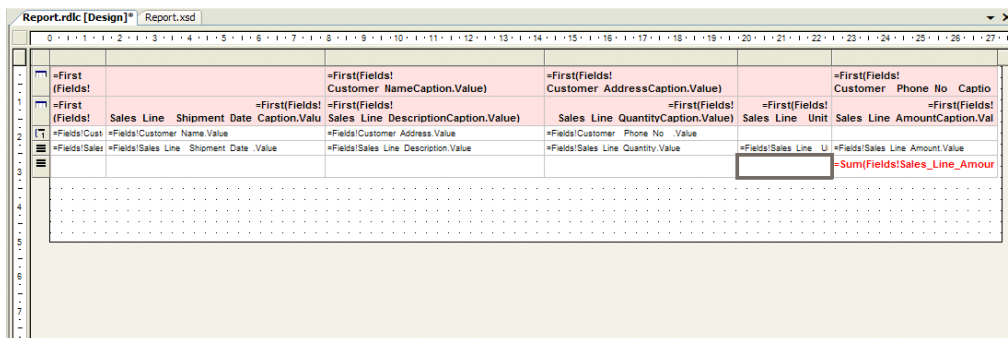
You can also easily add aggregate functions to your report in Visual Studio.

Add a row below the Sales Line row. In the last cell in this new row, add the following expression: `=Sum(Fields!Sales_Line_Amount.Value)`. This will calculate the total amount ordered by each customer.

Formatting and Style

You can easily format the appearance of many fields in your reports, including date and currency fields. In the **Properties** window, select the **Format** tab. Click the ellipse button and in the **Choose Format** window, select the format that you want the dates to appear in.

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:



When you are satisfied with the layout of the report, save and close the project.

In the Classic Client, when you open this report in the 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. Compile the report.

The database now contains two versions of this report – a C/SIDE version for the Classic Client and a .NET 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 you will convert the original C/SIDE version of the report again and won't be able to see the changes that you have already made.

Furthermore, you **must** remember that the C/SIDE Report Designer and Visual Studio are not closely integrated and that any changes made in one environment will not be visible in the other. You must save any changes that you make in either of the development environments and import or export the changes in the other environment before they will be visible there.

Web Services

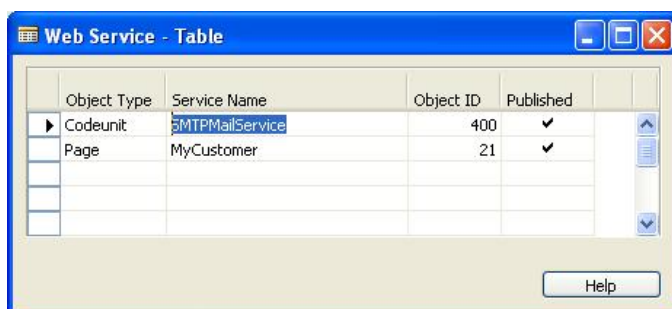
A Web service is a standardized way for independent software systems to communicate with each other. Web services are designed specifically to facilitate the highly dynamic data interchange that is required in business transactions. Standardized integration technologies — such as Web services — bring value to businesses by breaking down the de-facto “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.

Microsoft Dynamics NAV supports Web services out-of-the-box thereby making it easy to integrate Microsoft Dynamics NAV with other systems. Specifically, Web service integration with Microsoft Dynamics NAV is facilitated via *service-enabled* codeunits and pages. With proper authentication and authorization, external systems can read and write data on pages and call codeunits in the straightforward fashion 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 utilize 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, 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 in Microsoft Dynamics NAV

The way in which Microsoft Dynamics NAV integrates Web services spares developers the complex task of manually setting up Web service frameworks, for example, managing the WSDL descriptions (See sidebar). Publishing a particular page or codeunit as a Web service is a simple matter of adding it to the *Web Service* table. No additional steps are required.



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

The difference between Web Services and the World Wide Web

HTTP and HTML were designed to support interactive browsing of content that is generally static or at least highly cacheable. In contrast, the Web services architecture is designed for dynamic program-to-program interaction. In the Web services architecture, many kinds of distributed systems can be implemented. Examples of Web service 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 must be much more flexible than the first Web protocols. However, like the World Wide Web, Web services also rely on a small number of specific protocols, most prominently SOAP, Simple Object Access Protocol.

Web Service Descriptions

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) - an XML-based language for defining Web services – is used to express the interchange formats and message exchange patterns of the Web services.

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

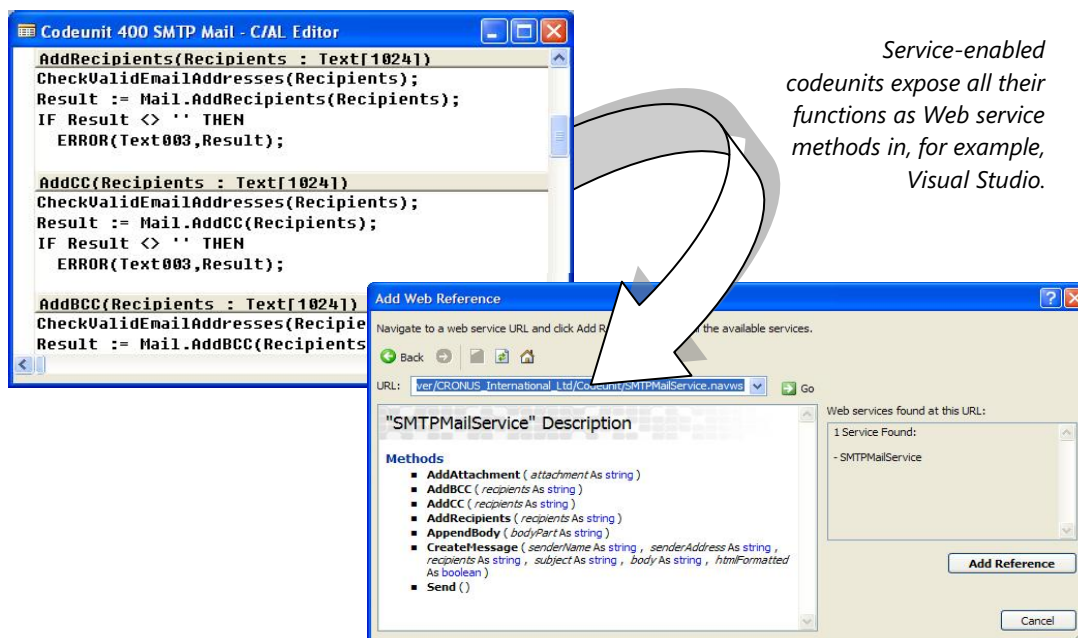
```
http://NAV_Server1/NavisionServer/CRONUS_International_Ltd/Page/MyCustomer.navws
```

It is important to know that Microsoft Dynamics NAV manages Web service requests in the same way as it handles requests from end-users. This means that the authorization and validation of access rights, the validation of input data and the invocation of business logic as well as 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 developers don't have to replicate code that validates data or invokes business logic when they build systems that use the Web services provided by Microsoft Dynamics NAV.

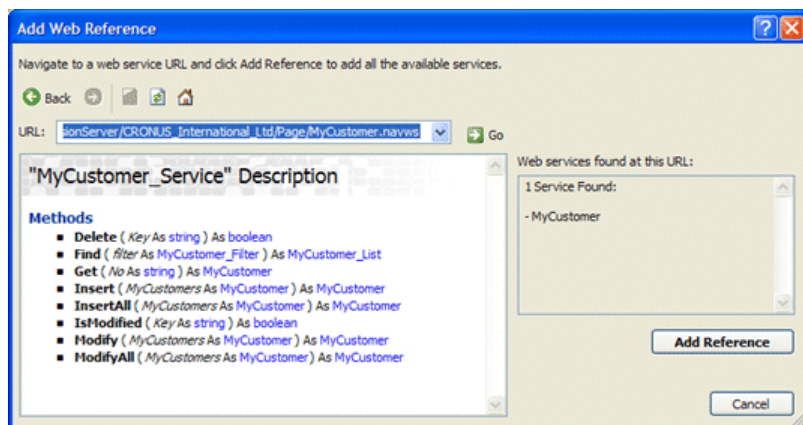
Working with Microsoft Dynamics NAV Web Services

Pages and codeunits that are published as Web services contain a certain set of *methods* that the consumers of these Web services can call to access the functions and the data that they contain.

When a Microsoft Dynamics NAV codeunit is published as a Web service, all the functions that it contains are made available as Web service methods.



When a Microsoft Dynamics NAV page is published as a Web service, a fixed set of 8 methods is made available to developers so that they can manage the common record handling operations, such as Insert, Modify, Find, and so on.



Service-enabled pages have a web service interface that contains methods for record handling, including filtering.

Service-enabling Codeunits that Expose Complex Data

A few extra steps are required when you need to Web service-enable codeunits with functions that use complex types in the input or return parameters. In this case, Dynamics NAV Web services depend on XMLport objects for the definition of the types of data that is exchanged with other systems.

Using XMLports - rather than records - for the definition of these data structures gives developers a quick and easy way to control the way data from Microsoft Dynamics NAV is available to external systems via Web services. For example, it is often useful to expose only a subset of the fields in the **Customer** table for a specific task. This not only helps keep data confidential, but also optimizes the performance of transactional systems by not exchanging unnecessary data.

XMLports are eminently suited to the task of providing complex data types for Web services because many of the entities or pieces of data that you might want to expose have a hierarchical structure. A typical example is the document type that contains a single header and multiple lines but there are many other complex types in a Dynamics NAV installation. For example, the Bill of Materials (BOM), customers with multiple ship-to addresses and even the product catalogues are all better suited to an XML-like structure rather than a classical .NET structure, such as a C# class.

Migrating to the Microsoft Dynamics NAV Role Tailored Client

The new Microsoft Dynamics NAV RoleTailored Client and Microsoft Dynamics NAV Portal for Microsoft SharePoint both run on Microsoft SQL Server. If you are running the Database Server for Microsoft Dynamics™ NAV Classic, then moving your system to the SQL Server Option is an essential part of migrating to the Microsoft Dynamics NAV RoleTailored Client.

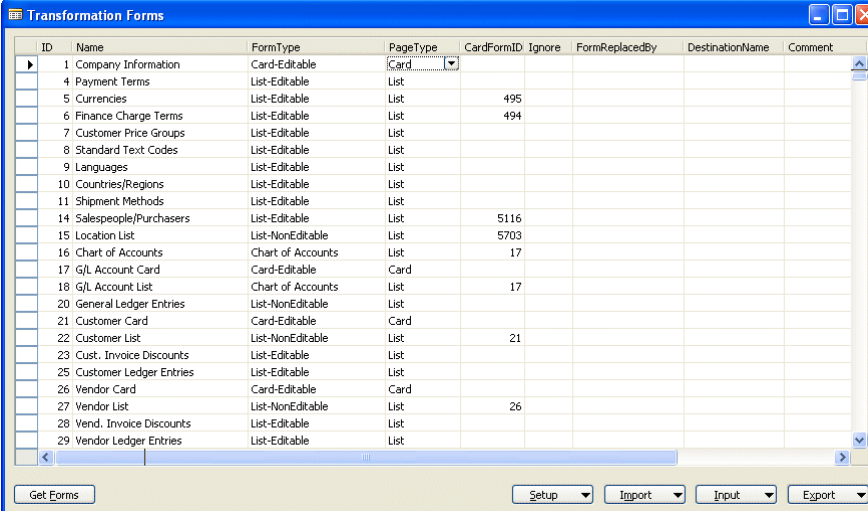
As mentioned earlier, in the Role Tailored Client and Portal for SharePoint, forms are replaced by pages. We have therefore developed a tool to help customers and partners create pages that contain the same functionality as the forms in their current application. This will help them migrate their application and all its functionality to the new architecture and take advantage of the features that it provides.

The Transformation Tool

The Transformation Tool has been designed to help you transform your Microsoft Dynamics NAV Classic application to one that runs on the new architecture by creating pages that contain the same functionality as your existing forms. The Transformation Tool creates a new page object for each form in your application. It does this 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.

The Transformation Tool is essentially a mapping tool that you use to map all the forms in your current application to the new page types that are supported by the new architecture. The tool is based on rules that we have developed for mapping forms to pages. Many of these rules have been hard coded into the tool and will generate new pages with controls and properties that correspond to particular new behavior available in the RoleTailored Client. Most of the new page objects that are created will contain the same functionality as the original forms.

One of the first steps in the transformation process is to map every form in the application to one of the new page types that have been defined for the new architecture. It is also possible to specify that certain forms should be ignored for a particular run of the Transformation Tool. Note you can run the Transformation tool as many times as you need to.



The screenshot shows the 'Transformation Forms' application window. It contains a table with the following columns: ID, Name, FormType, PageType, CardFormID, Ignore, FormReplacedBy, DestinationName, and Comment. The table lists various forms and their corresponding page types and IDs.

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					

When you run the tool it generates a log file that lists all of the forms that were transformed and contains details about what was done including logging details about what prevented it from successfully transforming a form that has an unrecognized structure. This information will be invaluable when you look at the forms again and need to decide whether or not they should be transformed. These forms will have to be redesigned before they can be transformed successfully. Alternatively you can ignore some or all of these forms and create new pages in the Page Designer that contain the same functionality.

One of the greatest strengths of Microsoft Dynamics NAV is the ease with which it allows you to customize the application. However, this flexibility has also meant that the upgrade process contains many manual steps as you must transfer/re-implement all these customizations every time you upgrade to a newer version of the product.

This is also the case when you migrate to the new architecture. The Transformation Tool can generate a page from any existing form. However the degree of success achieved by the Transformation Tool depends greatly on the extent and the type of customizations that you have carried out on any given form. The more the customizations correspond to the design of the standard application, the easier it will be to transform them successfully. Conversely, the more complex the customizations, the more difficult it will be to transform them.

Furthermore, we have changed some forms in the standard application to ensure that pages containing the same functionality can be easily generated by the Transformation Tool. To take advantage of the Transformation Tool's ability to automatically generate pages, you will probably have to re-customize a subset of forms to ensure that these pages can be generated successfully.

C/AL code now runs on the middle tier server rather than on the client computer and therefore additional changes may have to be made to your Dynamics NAV application before it can run on the new architecture. A simple example is the FILE.CREATE function. In the Classic Client, when code like this was run, files were created on the client. In the new architecture, the files are created on the Dynamics NAV Service. If the file is meant for a user, you must write extra code to download it to the client computer.

Pages contain fewer triggers than forms. There are two main reasons for this. Firstly, the RoleTailored Client has enhanced behavior hard coded into the controls that run on it – some code that used to be necessary has been made obsolete by these new controls. The second reason is that by taking advantage of the fact that we no longer need all the triggers, the performance of Pages can be increased by cutting down on communication between the Client and the Service.

The 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.

Careful preparation and the appropriate design changes will ensure that the Transformation Tool can successfully migrate your application to the new architecture.

Methodologies

When you decide to migrate an application to the new architecture, you must also decide which methodology is best suited to the application in question. Migrating to the new architecture can be an iterative process where you run the tool a number of times and carefully alter any problematic forms to ensure that pages can be created containing the required functionality.

It is important to remember that when you transform a form, it is not changed at all. For each form that you enter into the Transformation Tool, a new page object is created with matching functionality that can be used by the RoleTailored Client. You then have two objects – a form and a page. Furthermore, if you run the tool again and transform the same form, the new page that is created will overwrite the first page object.

If, at a later date, you decide that you want to change the functionality of your application, you can implement the changes in the form before transforming it again, implement the changes in the corresponding page or implement the changes in both objects, depending on how your application is used. Do you use the Classic Client (forms), the RoleTailored Client (pages) or do you run in mixed mode (both)?

In general, you can choose between the following migration methodologies:

Reactive Method

- Run the Transformation Tool
- Check the log file
- Redesign any forms that did not transform successfully – Matrix forms, wizards, and so on.
- Run the tool again to transform the forms that you have altered.

Proactive Method

- Redesign any forms that you know will not transform satisfactorily
- Run the Transformation Tool
- Check the log file
- Modify any forms that are still causing problems

Start from Scratch Method

- Implement your entire application in the new Page Designer and ignore the transformation tool.

Combined Method – all of the above

- Transform some forms to pages
- Create some new pages that contain your customizations
- Change some of your customizations and create new pages to get greater benefit from the new UI.

In short, the method you choose will probably vary from one implementation to another. A good development practice would be to use a copy of the database in question to test the waters and see which method or variation of methods achieves the best results for each implementation.

Features that have been Redesigned

A number of Microsoft Dynamics NAV features have been redesigned to ensure that they can be successfully transformed and run on the Dynamics NAV Service architecture:

Feature	Required Redesign
Files	<p>File functions used by C/AL are executed on the middle tier server and not locally on the client computer as is the case in the Classic Client. Therefore, when you write code for the application, you must remember where the files are created. Applications that generate files for the client to use must modify their code so that these files are downloaded to the RoleTailored Client. Similarly, they must upload files from the client so that they can be processed on the Service computer.</p> <p>New functions have been created to facilitate this file transfer. The new functions are:</p> <ul style="list-style-type: none"> • FILE.UPLOAD and FILE.UPLOADINTOSTREAM - are used to send a file from the RoleTailored Client to the Service computer. • FILE.DOWNLOAD and FILE.DOWNLOADINFROMSTREAM - are used to send a file from the Service computer to the RoleTailored Client.
Automation Objects	<p>There are four different ways of using Automation Objects in Dynamics NAV:</p> <p><i>1: Using a COM component to generate an Excel file for a user.</i></p> <p>C/AL code runs on the Service computer to CREATE the automation object and then SAVES the file on the Service computer. When the RoleTailored Client needs to use the file, the FILE.DOWNLOAD function sends it from the Service computer to the client computer.</p> <p>The COM DLL must be stored and registered on the Service computer.</p> <p><i>1: Using a COM component to carry out external processing (for example, call a Web Service)</i></p> <p>C/AL code runs on the Service computer to CREATE an automation object and then calls the method on the object. When the method is run, any return value is passed back to the running code.</p> <p>The COM DLL must be stored and registered on the Service Tier computer.</p> <p><i>3: Using a COM component to read from MSMQ message</i></p> <p>The RoleTailored Client runs code on the Service computer to CREATE an automation object and then READ from the message queue. The code is able to process the result of the READ.</p> <p>Note: If the location of the queue is specified locally (for example, “./myqueue”),</p>

	<p>the code will read from a message queue on the Service computer called "myqueue" and not from a message queue of the client computer.</p> <p><i>4: Using a visual OCX with its own window</i> You cannot use an OCX that has its own window (UI) to perform special tasks for the client.</p> <p>In the new architecture the client cannot run code or interact with other components that are also running on the client. 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 in turn delivers the information required by the system. You can also code actions to a page that reads from the database after the Web service has been called.</p>
Reports	<p>Reports are different on the RoleTailored Client than they are on the Classic Client. In the new architecture, the layout of the report is designed in Microsoft Visual Studio and totals are also defined in the layout editor in Microsoft Visual Studio. Furthermore, because the layout is designed in Microsoft Visual Studio Microsoft Dynamics NAV 5.1 ignores all the code on section triggers</p> <p>If you have a report that you want to use on the Service Architecture, you must transform the report and redesign its layout in Microsoft Visual Studio. You should also consider moving any special code that you have written for a report from the section triggers to another section in the report or to a codeunit.</p>
Batch Jobs/ Processing-only Reports	<p>Processing-only reports, also known as batch jobs, require the same work as the ordinary reports that we have just described. However, since these reports are also commonly saved to file or printed to file after they have been run, the comments that we made earlier about files also apply to these reports.</p>
Forms	<p><i>Statistics Forms</i> 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 like a matrix and the all the cells in the page must be filled. You must add a placeholder control to empty cells to fill the space.</p> <p>Recommendation: Consider redesigning some of the current statistics forms as reports because statistics forms are generally used for data analysis.</p> <p><i>Forms with Info Frames and Forms with Filter frames</i> Forms that contain Info Frames (e.g. the Item Charge Assignment form or Cash Receipt Journal) or Forms with Filter frames (e.g. the Sales Prices 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 not redesigned the default 2-column layout of the RoleTailored Client is used, so in some instances you may prefer to rearrange the fields to obtain a more user-friendly order of field in the RoleTailored Client.</p> <p><i>Matrix Forms</i> The Microsoft Dynamics NAV 5.1 Role Tailored 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 limitations:</p>

	<p>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/matrix forms in Microsoft Dynamics NAV 5.1 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.</p> <p><i>Journals and worksheets with batches</i> Journals and worksheets with batches are now accessed through a List Place (of batches). So adding an "Edit Journal" command to the batch form is required. Navigation Panes on RoleCenters should refer to new forms (List Places)</p> <p><i>Type-specific filtering on forms</i> Role-based Client does not support filtering in the Department Page (Menu Suite). So added new forms (Lists) for "Type" Specific filtering is required, e.g. Sales List split into 6 new Lists specific to the Document Type.</p> <p><i>Wizards</i> In Microsoft Dynamics NAV 5.0, wizards are supported in a single form. Wizards must be redesigned and a new page constructed for each "window" of the wizard before they can run on the Service Architecture.</p>
Display Properties in Forms	<p>None of the properties that govern fonts – FontSize, FontItalic and so on - are supported in the new architecture. These properties are ignored by the RoleTailored Client because the behavior of the UI is controlled by the client.</p> <p>The Visible, Editable and Enabled properties have also been changed. In Microsoft Dynamics NAV forms, you can set the Boolean value of the property from code. In pages, the value is defined in a property of the control but it is also possible to set the value from an expression. For example, you can set the value to be true if a certain value is present in a variable or another control has a particular value.</p>
Infinitely long running processes	<p>The Microsoft Dynamics NAV Service 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.</p> <p>To build a similar solution, the application must take a more event driven approach rather than a polling/always on approach. For example, you can implement most solutions using Web services as the event trigger.</p> <p>Consider changing your solution to a Web services solution. This will also increase interoperability and reliability.</p>

Service Tier Architecture and C/SIDE side by side

The Dynamics NAV C/SIDE client and its related cousin, the Dynamics NAV Application Server are able to run on the same SQL database as is used by the Service Tier Architecture. Such a system allows Role Tailored Clients and C/SIDE clients to be run within a system simultaneously offering different users access to the same application. This mix of clients running on the same system is referred to Mixed Mode. The clearest benefits of such a system are not only that different users in an organization are able to use different clients but also that any specific solutions that are not immediately migrated to the Role Tailored Client may still be used within an organization and also any integration solutions running on the NAV Application Server will continue to function as well.

Mixed Mode however introduces constraints when using Dynamics NAV and such it should be judged carefully before a decision is taken to deploy such a solution. This section of the paper seeks to raise the issue of mixed mode and enable you to decide if the benefits of the mixed mode system outweigh the costs.

You will probably have observed at this point that this entire paper is describing an actual mixed mode system. It is correct that using the C/SIDE client as the development environment attached to a Service Tier Architecture is an example of mixed mode. This scenario is supported for two main reasons – firstly that it is the C/SIDE Client that offers the development environment. It simply must be supported in order to allow any kind of accelerated development environment where a programmer can see a change they have made in the application be applied to the running system. The second reason is that this scenario is simpler from other platform considerations – specifically it is not necessary to enforce the Dynamics NAV security roles and permissions when working at design time.

The more commonly expected scenario is that of different end users running different types of clients. The major drawbacks in trying to support a mixed mode system centre mostly on increased complexity during development for the partner whose role it is to develop and maintain the system.

The two main issues are:

- Application code and Functionality
- Security

Application Code, Application Functionality and Testing

As described earlier in this document, there are differences between the role tailored client and the C/SIDE client. As the application is defined in code and/or properties from the development environment, it becomes necessary to write code specifically for one platform or another. Writing code that targets particular features of each of the platforms results in needing to write (and maintain!) code that looks like this:

```
IF (ISSERVICETIER) THEN
  ... <service tier code>
ELSE
  ... <C/SIDE code>
ENDIF
```

Writing for specific platforms in such a structure vastly reduces the ease of understanding and maintenance of the code. In particular, reports are different between the two platforms, as is code that uses Automation objects and Files. Another example of application functionality that is different between the C/SIDE and the Role Tailored Client is the Office Integration Stylesheets. Continuing to

develop for the stylesheets feature will require not only double work, but a more complex understanding of how particular stylesheets should only be applied to one specific platform type .

Both client platforms use the menusuite object for navigation but the design is different. In the C/SIDE Client, a user opens a card and then can jump to a list. In the Role Tailored Client, the user is presented with a list and can then select which Page (card) to open. Supporting both these types of designs in a single application becomes double work to understand and maintain changes.

C/SIDE Client and Role Tailored Client have different implementations of the feature used to record miscellaneous notes or comments about a document or contact. The C/SIDE Client uses the *comments* feature and the Role Tailored Client uses *notes*. Both provide similar functionality but different users are not able to see the information entered by another user who uses a different client.

Writing code to run correctly on both platforms will require double the level of testing

Security

Dynamics NAV 4.0 introduced the Enhanced Security Model for the SQL Option for C/SIDE Clients. In this version each user is assigned an SQL Application Role rather than having a single Application Role for the whole system.

The Dynamics NAV NST architecture uses the previous single Application Role for enforcing Security. This is permissible as the Role Tailored Client enforces security and trust on the client and in the protocol that interacts with the NST. It does mean that any C/SIDE clients in a mixed mode environment will not be able to run with the Enhanced Security Model.

Reports

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

Report Properties:

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

Data Item Properties:

NewPagePerGroup
NewPagePerRecord
TotalFields
GroupTotalFields

Report Functions:

NewPagePerRecord
URL
ObjectID

CurrReport Functions:

CreateTotals
TotalsCausedBy
ShowOutput
PageNo
NewPage
SaveAsXML
PaperSource
NewPagePerRecord
URL
ObjectID

Report Triggers/Functions:

Section triggers

Layout-specific functions and totaling functions
Unsupported functions referred to from other objects (CodeUnits, Tables, etc.)

The information contained in this document represents the current view of Microsoft Corporation on the issues discussed as of the date of publication. Because Microsoft must respond to changing market conditions, this document should not be interpreted to be a commitment on the part of Microsoft, and Microsoft cannot guarantee the accuracy of any information presented after the date of publication.

This White Paper is for informational purposes only. MICROSOFT MAKES NO WARRANTIES, EXPRESS, IMPLIED, OR STATUTORY, AS TO THE INFORMATION IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Microsoft Corporation.

Microsoft may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Microsoft, the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.

© 2007 Microsoft Corporation. All rights reserved.

Microsoft, The Microsoft Dynamics Logo, Microsoft Navision, Microsoft SQL Server, are either registered trademarks or trademarks of Microsoft Corporation or Microsoft Business Solutions ApS in the United States and/or other countries. Microsoft Business Solutions ApS is a subsidiary of Microsoft Corporation.

Microsoft