NAV 2015 is a complete ERP system, which also contains a robust set of development tools to support customization and enhancement. These include an object designer for each of seven application object types, a business application-oriented programming language with .NET interface capability, a compiler, a debugger, and programming testing language support.

This book is designed to take you from an introduction to the product and its integrated development tools to being a productive developer in the NAV 2015 environment. It will serve as a comprehensive reference guide, complementing NAV’s Help files. You will find this book really useful if you want to evaluate the product’s development capabilities or need to manage NAV 2015 based projects. Additionally, you will also learn about the NAV application structure, the C/SIDE development environment, the C/AL language, the construction and uses of each object type, and how it all fits together.

Who this book is written for

This book is targeted at those who want to learn NAV’s powerful and extensive built-in development capabilities. It assumes that you understand programming and are familiar with business application software.

What you will learn from this book

- Extract and utilize NAV data using queries and reports.
- Use the integrated debugger, the .NET interface, and C/AL extensions for automated testing.
- Understand NAV’s data element choices, properties, and usages.
- Discover how to successfully deliver an interactive user data presentation.
- Sharpen your skills and increase your productivity with Microsoft Dynamics NAV.
- Get to grips with the fundamental programming and design concepts of Microsoft Dynamics NAV 2015.

In this package, you will find:

- The author's biography
- A preview chapter from the book, Chapter 1 'An Introduction to NAV 2015'
- A synopsis of the book’s content
- More information on Programming Microsoft Dynamics™ NAV 2015
About the Authors

**David Studebaker** is the Chief Technical Officer and a cofounder of Liberty Grove Software with his partner, Karen Studebaker. David has had a wide range of development, consulting, sales and management roles throughout his career. He has been a partner or owner and manager of several software development businesses while always maintaining a significant role as a business application developer.

David started programming in 1962. He has been developing in C/AL since 1996. David has been an active participant in each step of computing technology from the first solid state mainframes to today's technology, from binary assembly language coding to today's C/AL. David's special achievements include his role as co-developer of the first production multiprogrammed SPOOLing system in 1967. He has worked on a diverse set of software applications including manufacturing, distribution, retail, engineering, and others.

David has a BS in mechanical engineering from Purdue University and an MBA from the University of Chicago. He has been writing for publications since he was an undergraduate and has had numerous magazine and reference books published. Prior to coauthoring this book, David was the author of *Programming Microsoft Dynamics NAV* (for the Classic Client), *Programming Microsoft Dynamics NAV 2009* (for the Role Tailored Client), and *Programming Microsoft Dynamics NAV 2013*. He has been a member of the Association for Computing Machinery since 1963 and was a founding officer of two local chapters of the ACM.
Christopher Studebaker is an NAV developer/implementer and has 15 years' experience in designing, developing, implementing, and selling in the NAV and SQL Server environments. He has specialized in retail, manufacturing, job shop, and distribution implementations, mostly in high user count and high data volume applications. Chris has worked on many NAV implementations with integrations to external databases and third-party add-on products. Some special applications include high-volume order entry, pick-to-light systems, procurement analysis, and web frontends.

Chris acts in a consulting and training role for customers and for peer NAV professionals. He provides training both in informal and classroom situations, often developing custom course material to support courses tailored to specific student group needs. Courses have included various NAV functional and development areas.

Before becoming a certified NAV developer, Chris was a certified environmental consultant working with manufacturing facilities to meet national and state regulations. His duties included regulatory reporting, data analysis, project management, and subcontractor oversight. His accomplishments include obtaining several safety certifications and managing projects for hazardous material management and abatement.

Chris is an expert at NAV installation, configuration, and development. He has been working with SQL in both NAV and other Microsoft applications for over a decade. He has a bachelor of science degree from Northern Illinois University and has done graduate work at Denmark Technical University. Chris was the coauthor of the Packt Publishing book Programming Microsoft Dynamics 2013.
Welcome to the worldwide community of Microsoft Dynamics NAV developers. This is a collegial environment populated by C/AL developers who readily and generously share their knowledge. There are formal and informal organizations of NAV-focused users, developers, and vendor firms scattered around the globe and active on the Web. Our community continues to grow and prosper, now including over 110,000 user companies worldwide.

The information in this book will help you to shorten your learning curve of how to program for the NAV 2015 ERP system using the C/AL language, the C/SIDE integrated development environment and their capabilities. We hope you enjoy working with NAV as much as we have.

**A brief history of NAV**
Each new version of Microsoft Dynamics NAV is the result of inspiration and hard work along with some good fortune and expert technical investment over the last thirty years.

**The beginning**
Three college friends, Jesper Balser, Torben Wind, and Peter Bang, from Denmark Technical University (DTU) founded their computer software business in 1984 when they were in their early twenties. This business was Personal Computing & Consulting (PC & C) and its first product was called PC Plus.
Single user PC Plus
PC Plus was released in 1985 with the primary goal of ease of use. An early employee said its functional design was inspired by the combination of a manual ledger journal, an Epson FX 80 printer, and a Canon calculator. Incidentally, Peter Bang is the grandson of one of the founders of Bang & Olufsen, the manufacturer of home entertainment systems par excellence.

PC Plus was a PC DOS-based single user system. PC Plus' design features included these:

- An interface resembling the use of documents and calculators
- Online help
- Good exception handling
- Minimal computer resources required

The PC Plus product was marketed through dealers in Denmark and Norway.

The multi-user Navigator
In 1987, PC & C released a new product, the multi-user Navigator and a new corporate name, Navision. Navigator was quite a technological leap forward. It included the following:

- Client/Server technology
- A relational database
- Transaction-based processing
- Version management
- High-speed OLAP capabilities (SIFT technology)
- A screen painter tool
- A programmable report writer

In 1990, Navision was expanding its marketing and dealer recruitment efforts in Germany, Spain, and the United Kingdom. Also in 1990, V3 of Navigator was released. Navigator V3 was still a character-based system, albeit a very sophisticated one. If you get an opportunity to study Navigator V3.x, you would instantly recognize the roots of today's NAV product. By V3, the product included these features:

- A design based on object-oriented concepts
- Integrated 4GL Table, Form, and Report Design tools (the IDE)
- Structured exception handling
• Built-in resource management
• The original programming language that became C/AL
• Function libraries
• The concept of regional or country-based localization

When Navigator V3.5 was released, it also included support for multiple platforms and databases. Navigator V3.5 would run on both Unix and Windows NT networks. It supported the Oracle and Informix databases, as well as the one developed in-house.

Around this time, several major strategic efforts were initiated. On the technical side, the decision was made to develop a GUI-based product. The first prototype of Navision Financials (for Windows) was shown in 1992. At about the same time, a relationship was established that would take Navision into distribution in the United States. The initial release in the US in 1995 was V3.5 of the character-based product, rechristened as Avista for the US distribution.

**Navision Financials for Windows**

In 1995, Navision Financials V1.0 for Microsoft Windows was released. This product had many (but not all) of the features of Navigator V3.5. It was designed for complete look-and-feel compatibility with Windows 95. There was an effort to provide the ease of use and flexibility of development of Microsoft Access. The new Navision Financials was very compatible with Microsoft Office and was thus sold as "being familiar to any Office user". Like any V1.0 product, it was quickly followed by a much improved V1.1.

In the next few years, Navision continued to be improved and enhanced. Major new functionalities were added, such as:

• Contact Relation Management (CRM)
• Manufacturing (ERP)
• Advanced Distribution (including Warehouse Management)

Various Microsoft certifications were obtained, providing muscle to the marketing efforts. Geographic and dealer base expansion continued apace. By 2000, according to the Navision Annual Report of that year, the product was represented by nearly 1,000 dealers (Navision Solution Centers) in 24 countries and used by 41,000 customers located in 108 countries.
Growth and mergers
In 2000, Navision Software A/S and its primary Danish competitor, Damgaard A/S, merged. Product development and new releases continued for the primary products of both original firms (Navision and Axapta). In 2002, the now much larger Navision Software, with all its products (Navision, Axapta, and the smaller, older C5, and XAL) was purchased by Microsoft, becoming part of the Microsoft Business Systems division along with the previously purchased Great Plains Software business and its several product lines. All the Navision and Great Plains products received a common rebranding as the Dynamics product line. Navision was renamed Dynamics NAV.

Continuous enhancement
As early as 2003, research began with the Dynamics NAV development team planning moves to further enhance NAV and take advantage of various parts of the Microsoft product line. Goals were defined to increase integration with products such as Microsoft Office and Microsoft Outlook. Goals were also set to leverage the functional capabilities of Visual Studio and SQL Server, among others. All the while, there was a determination not to lose the strength and flexibility of the base product.

NAV 2009 was released in late 2008, NAV 2013 in late 2012, followed by NAV 2015 in late 2014. The biggest hurdles to the new technologies have been cleared. A new user interface, the Role Tailored Client, was created as part of this renewal. NAV was tightly integrated with Microsoft's SQL Server and other Microsoft products such as Office, Outlook, and SharePoint. Development is more integrated with Visual Studio and more .NET compliant. The product is becoming more open and, at the same time, more sophisticated supporting features such as Web Services access, Web and tablet clients, the integration of third-party controls, RDLC, and Word-based reporting, and so on.

Microsoft continues to invest in, enhance, and advance NAV. More new capabilities and features are yet to come, continuing to build on the successes of the past. We will all benefit.

C/AL's Roots
One of the first questions asked by people new to C/AL is often "what other programming language is it like?" The best response is "Pascal". If the questioner is not familiar with Pascal, the next best response would be "C" or "C#".
At the time the three founders of Navision were attending classes at Denmark Technical University (DTU), Pascal was in wide use as a preferred language not only in computer courses, but also in other courses where computers were tools and software had to be written for data analyses. Some of the strengths of Pascal as a tool in an educational environment also served to make it a good model for Navision's business applications development.

Perhaps coincidentally (perhaps not) at DTU in this same time period, a Pascal compiler called Blue Label Pascal was developed by Anders Hejlsberg. That compiler became the basis for what was Borland's Turbo Pascal, which was the "everyman's compiler" of the 1980s because of its low price. Anders went with his Pascal compiler to Borland. While he was there, Turbo Pascal morphed into the Delphi language and the IDE tool set under his guidance.

Anders later left Borland and joined Microsoft, where he led the C# design team. Much of the NAV-related development at Microsoft is now being done in C#. So the Pascal-C/AL-DTU connection has come full circle, only now it appears to be C#-C/AL. Keeping it in the family, Anders' brother, Thomas Hejlsberg also works at Microsoft on NAV as a Software Architect. Each in their own way, Anders and Thomas continue to make significant contributions to Dynamics NAV.

In a discussion about C/AL and C/SIDE, Michael Nielsen of Navision and Microsoft, who developed the original C/AL compiler, runtime, and IDE, said that the design criteria were to provide an environment that could be used without the following:

- Dealing with memory and other resource handling
- Thinking about exception handling and state
- Thinking about database transactions and rollbacks
- Knowing about set operations (SQL)
- Knowing about OLAP (SIFT)

Paraphrasing some of Michael's additional comments, the goals of the language and IDE design were to do the following:

- Allow the developer to focus on design, not coding, but still allow flexibility
- Provide a syntax based on Pascal, stripped of complexities, especially relating to memory management
- Provide a limited set of predefined object types, reduce the complexity and learning curve
• Implement database versioning for a consistent and reliable view of the database
• Make the developer and end user more at home by borrowing a large number of concepts from Office, Windows, Access, and other Microsoft products

Michael is still working as part of the Microsoft team in Denmark on new capabilities for NAV. This is another example of how, once part of the NAV community, most of us want to stay part of this community.

What you should know
To get the maximum out of this book as a developer, you should have the following attributes:

• Be an experienced developer
• Know more than one programming language
• Have IDE experience
• Be knowledgeable about business applications
• Be good at self-directed study

If you have these attributes, this book will help you become productive with C/AL and NAV much more rapidly.

Even though this book is targeted first at developers, it is also designed to be useful to executives, consultants, managers, business owners, and others who want to learn about the development technology and operational capabilities of Dynamics NAV. If you fit into one of these or similar categories, start by studying *Chapter 1, An Introduction to NAV 2015*, for a good overview of NAV and its tools. Then you should review sections of other chapters as the topics apply to your specific areas of interest.

This book’s illustrations are from the W1 Cronus database Dynamics NAV V2015.

What this book covers
*Chapter 1, An Introduction to NAV 2015*, starts with an overview of NAV as a business application system. This is followed by an introduction to the seven types of NAV objects, and the basics of C/AL and C/SIDE. Then we will do some hands-on work and define Tables, multiple Page types, and a Report. We'll close with a brief discussion of how backups and documentation are handled in C/SIDE.
Chapter 2, *Tables*, focuses on the foundation level of NAV data structure: Tables and their structures. We will cover Properties, Triggers (where C/AL resides), Field Groups, Table Relations, and SumIndexFields. We'll work our way through the hands-on creation of several tables in support of our example application. We will also review the types of tables found in the NAV applications.

Chapter 3, *Data Types and Fields*, we will learn about fields, the basic building blocks of the NAV data structure. We review the different Data Types in NAV. We will cover all the field properties and triggers in detail. We'll also review the three different Field Classes. We'll conclude with a discussion about the concept of filtering and how it should be considered in the database structure design.

Chapter 4, *Pages – The Interactive Interface*, we will review the different types of pages, their structures (Triggers, Properties) and general usage. We'll build several pages for our example application using Page Wizard and Page Designer. We will also study the different types of controls that can be used in the pages. In addition, we'll review how and where actions are added to the pages.

Chapter 5, *Queries and Reports*, we will learn about both Queries and Reports, two methods of extracting data for presentation to users. For Queries, we will study how they are constructed and some of the ways they are utilized. For Reports, we will walk through report data flow and the variety of different report types. We will study the two Report Designers, the C/SIDE Report Designer and the Visual Studio Report Designer and how a NAV report is constructed using both of these. We'll learn what aspects of reports use one designer and what aspects use the other. As in the previous studied objects, we will discuss Properties and Triggers. We will review how reports can be made interactive and will do some hands-on report creation.

Chapter 6, *Introduction to C/SIDE and C/AL*, we will learn about general Object Designer Navigation as well as the individual Designers (Table, Page, Report). We'll study C/AL code construction, syntax, variable types, expressions, operators, and functions. We will then take a closer look at some of the more frequently used built-in functions. The chapter will wrap up with an exercise on adding some C/AL code to a report objects created in an earlier exercise.

Chapter 7, *Intermediate C/AL*, we will dig deeper into C/AL development tools and techniques. We will review some more advanced built-in functions including those relating to dates and decimal calculations, both critical business application tools. We'll study the C/AL functions that support process flow control functions, input/output, and filtering. Then we'll do a review of methods of communication between objects. Finally, we'll apply some of what we've learned to enhance our example application.
Chapter 8, *Advanced NAV Development Tools*, we will review some of the more important elements of the Role Tailored User Experience, in particular the Role Center Page construction. We will dig into the components of a Role Center Page and how to build one. We’ll also cover XMLports and Web Services, two of the powerful ways of connecting NAV applications to the world outside of NAV. To better understand these, we will not only review their individual component parts, but also go through the hands-on effort of building an example of each one.

Chapter 9, *Successful Conclusions*, we will study in detail how NAV functions are constructed and learn how to construct your own functions. We will learn more about tools and features built into C/AL and C/SIDE. We will study the new debugger, review the support for Test-Driven Development, and take a look at the ability to integrate .NET Client Add-ins. We will integrate a .NET Add-in into our example applications. Finally, we will review tips to design efficiently, update and upgrade the system with the goal of helping us to become more productive and high quality NAV developers.

*Appendix, Review Answers*, provides you with the answers to the questions given in each chapter.
An Introduction to NAV 2015

“All truths are easy to understand once they are discovered; the point is to discover them.”

– Galileo Galilei

“Everything really interesting that happens in software projects eventually comes down to people.”

– James Bach

Microsoft Dynamics NAV has one of the largest installed user bases of any enterprise resource planning (ERP) system serving over 100,000 companies and one million plus individual users. The community of supporting organizations, consultants, implementers, and developers continues to grow and prosper. The capabilities of the off-the-shelf product increase with every release. The selection of the add-on products and services expands both in variety and depth.

The release of Microsoft Dynamics NAV 2015 continues its 20 plus year history of continuous product improvement. It provides more user options for access and output formatting. For new installations, NAV 2015 includes tools for rapid implementation. For all installations, it provides enhanced business functionality and more support for ERP computing in the cloud, including integration with Office 365. In addition, a new approach to upgrading that comes with NAV 2015 promises to lower the cost of ownership in the future.
Our goal in this chapter is to gain a big picture understanding of NAV 2015. You will be able to envision how NAV can be used by the managers (or owners) of an organization to help manage activities and the resources, whether the organization is for-profit or not-for-profit. You will also be introduced to the technical side of NAV from a developer's point of view.

In this chapter, we will take a look at NAV 2015, including the following:

- A general overview of NAV 2015
- A technical overview of NAV 2015
- A hands-on introduction to Client/Server Integrated Development Environment (C/SIDE) development in NAV 2015

**NAV 2015 – an ERP system**

NAV 2015 is an integrated set of business applications designed to service a wide variety of business operations. Microsoft Dynamics NAV 2015 is an ERP system. An ERP system integrates internal and external data across a variety of functional areas, including manufacturing, accounting, supply chain management, customer relationships, service operations, human resources management, as well as the management of other valued resources and activities. By having many related applications well integrated, a full featured ERP system provides an enter data once, use many ways information processing toolset.

NAV 2015 ERP addresses many functional areas. Some of them are listed as follows:

- Basic accounting functions (for example, general ledger, accounts payable, accounts receivable)
- Order processing and inventory (for example, sales orders, purchase orders, shipping, inventory, receiving)
- Relationship management (for example, vendors, customers, prospects, employees, contractors)
- Planning (for example MRP, sales forecasting, production forecasting)
- Other critical business areas (for example, manufacturing, warehouse management, marketing, cash management, fixed assets)

A good ERP system such as NAV 2015 is modular in design, which simplifies implementation, upgrading, modification, integration with third-party products, and expansion for different types of clients. All the modules in the system share a common database and, where appropriate, common data.
The groupings of individual NAV 2015 functions based on the department's menu structure is shown in the following figure. It is supplemented by information from Microsoft marketing materials and some of the groupings are a bit arbitrary. The important thing is to understand the overall components that make up the NAV 2015 ERP system.

NAV 2015 has two quite different styles of user interface (UI). One UI, the Development Environment, targets developers. The other UI style, the RoleTailored Client, targets end users. In NAV 2015, there are three instances of the RoleTailored Client – for Windows, for Web interaction, and for tablet use. The example images in the following module descriptions are from the RoleTailored Client Departments menu in the Windows Client.
Financial Management

Financial Management is the foundation of any ERP system. No matter what the business is, the money must be kept flowing, and the flow of money must be tracked. The tools which help to manage the capital resources of the business are part of NAV 2015's Financial Management module. These include all or part of the following application functions:

- **General Ledger**: Manages overall finances of the firm
- **Cash Management and Banking**: Manages inventory of money
- **Accounts Receivable**: Tracks the incoming revenue
- **Accounts Payable**: Tracks the outgoing funds
- **Analytical Accounting**: Analyzes the various flows of funds
- **Inventory and Fixed Assets**: Manages the inventories of goods and equipment
- **Multi-Currency and Multi-Language**: Supports international business activities

The **Financial Management** section of the **Departments** menu looks as follows:

<table>
<thead>
<tr>
<th>Financial Management</th>
<th>Payables</th>
</tr>
</thead>
<tbody>
<tr>
<td>General Ledger</td>
<td></td>
</tr>
<tr>
<td>Cash Management</td>
<td>Fixed Assets</td>
</tr>
<tr>
<td>Cost Accounting</td>
<td>Inventory</td>
</tr>
<tr>
<td>Cash Flow</td>
<td>Periodic Activities</td>
</tr>
<tr>
<td>Receivables</td>
<td>Setup</td>
</tr>
</tbody>
</table>

Manufacturing

NAV 2015 **Manufacturing** is general purpose enough to be appropriate for **Make to Stock (MTS)**, **Make to Order (MTO)**, **Assemble to Order (ATO)**, as well as various subsets and combinations of those. Although off-the-shelf NAV is not particularly suitable for most process manufacturing and some of the very high volume assembly line operations, there are third-party add-on and add-in enhancements available for these applications. As with most of the NAV application functions, Manufacturing can be implemented to be used in a basic mode or as a full featured system.
NAV Manufacturing includes the following functions:

- **Product Design (BOMs and Routings):** Manages the structure of product components and the flow of manufacturing processes
- **Capacity and supply requirements planning:** Tracks the intangible and tangible manufacturing resources
- **Production scheduling (infinite and finite):** Execution and tracking quantities and costs, plus tracking the planned use of manufacturing resources, both on an unconstrained and constrained basis

The **Manufacturing** section of the **Departments** menu looks as follows:

<table>
<thead>
<tr>
<th>Manufacturer</th>
<th>Execution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Product Design</td>
<td>Costing</td>
</tr>
<tr>
<td>Capacities</td>
<td>Planning</td>
</tr>
</tbody>
</table>

**Supply Chain Management**

Obviously, some of the functions categorized as part of NAV 2015 **Supply Chain Management (SCM),** for example sales and purchasing, are actively used in almost every NAV implementation. The supply chain applications in NAV include all or parts of the following applications:

- **Sales order processing and pricing:** Supports the heart of every business
- **Purchasing (including requisitions):** Includes planning, entering, pricing, and processing purchase orders
- **Inventory Management:** Manages inventories of goods and materials
- **Warehouse management including receiving and shipping:** Manages the receipt, storage, retrieval, and shipment of material and goods in warehouses
Even though we might consider **Assembly** to be part of **Manufacturing**, the standard NAV 2015 **Departments** menu includes it in the **Warehouse** section. The **Supply Chain Management** section of the **Departments** menu looks as follows:

```
Sales & Marketing
  Sales
  Order Processing
Purchase
  Planning
  Order Processing
Warehouse
  Orders & Contacts
  Planning & Execution
  Goods Handling Order b...
```

As a whole, these functions constitute the base components of a system appropriate for distribution operations, including those which operate on an Assemble to Order basis.

**Business Intelligence and reporting**

Although Microsoft marketing materials identify **Business Intelligence (BI)** and reporting as though it were a separate module within NAV, it's difficult to physically identify it as such. Most of the components used for BI and reporting purposes are appropriately scattered throughout various application areas. In the words of one Microsoft document, "Business Intelligence is a strategy, not a product." Functions within NAV that support a BI strategy include the following:

- **Standard Reports**: These are distributed ready-to-use by end users
- **Account schedules and analysis reports**: These are a specialized report writer for General Ledger data
- **Query, XMLport, and Report Designers**: These are developer tools to support the creation of a wide variety of report formats, charts, XML, and CSV files
- **Analysis by dimensions**: This is a capability embedded in many of the other tools
Chapter 1

- **Interfaces into Microsoft Office and Microsoft Office 365 including Excel**: These support the communications of data either into NAV or out of NAV
- **RDLC report viewer**: This provides the ability to present NAV data in a variety of textual and graphic formats; includes user interactive capabilities
- **Interface capabilities such as DotNet Interoperability and Web Services**: These are the technologies to support interfaces between NAV 2015 and external software products

**Relationship Management**

NAV’s **Relationship Management (RM)** functionality is definitely the *little sister* (or, if you prefer, *little brother*) of the fully featured standalone Microsoft **Customer Relationship Management (CRM)** system. The big advantage of the NAV RM is its tight integration with NAV customer and sales data. For those who need the full Microsoft CRM, prior versions of NAV have had a module connecting it to NAV. The same connector has been released for NAV 2015.

Also falling under the heading of CRM is the NAV **Service Management (SM)** functionality. While the RM component shows up in the menu as part of sales and marketing, the SM component is identified as an independent function in the menu structure.

- NAV functions that support RM are as follows:
  - **Marketing campaigns**: Plans and manages promotions
  - **Customer activity tracking**: Analyzes customer orders
  - **To do lists**: Manages what is to be done and tracks what has been done

- NAV functions that support SM are as follows:
  - **Service contracts**: Supports service operations
  - **Labor and part consumption tracking**: Tracks the resources consumed by the service business
  - **Planning and dispatching**: Manages service calls
Human Resource management

NAV Human Resources (HR) is a very small module, but relates to a critical component of the business - employees. Basic employee data can be stored and reported via the master table (in fact, one can use HR to manage data about individual contractors in addition to employees). A wide variety of individual employee attributes can be tracked by the use of dimensions fields. NAV functions that support HR are as follows:

- **Employee tracking**: Maintains basic employee description data
- **Skills inventory**: Maintains an inventory of the capabilities of the employees
- **Absence tracking**: Maintains basic attendance information
- **Employee statistics**: Tracks government required employee attribute data such as age, gender, length of service, and so on

Project Management

The NAV project management module consists of the jobs functionality supported by the resources functionality. Projects can be short or long term. They can be external (in other words - billable) or internal. This module is often used by third-parties as the base for vertical market add-ons (such as construction or job-oriented manufacturing). This application area includes parts or all of the following functions:

- **Budgeting and cost tracking**: Manages project finances
- **Scheduling**: Plans project activities
- **Resource requirements and usage tracking**: Manages people and equipment
- **Project accounting**: Tracks the results

Significant changes in NAV 2015

NAV 2015 contains added capabilities in a variety of areas including business functionality, enhanced development tools, a tablet client, more Internet compatibility, and increased integration to other Microsoft products. For information on what was new in NAV 2013 R2, review the MSDN notes at:


Some of the mentioned items include Manual Payment Processing, bank reconciliation enhancement, SEPA Debit and Credit handling, Web Client Enhancements, new Windows PowerShell capabilities, a new Help Server, and features to better implement and manage NAV on Windows Azure.
Information on the minimum hardware and software requirements to install and run Microsoft Dynamics NAV 2015 are found in the embedded Developer and IT Pro Help topic System Requirements for Microsoft Dynamics NAV 2015. Following are some of the specific areas where NAV 2015 contains significant changes (this list is representative, not comprehensive).

**Application changes**

Significant changes to applications include:

- Mandatory indicator fields (required data elements)
- Auto-fill and hidden fields
- Document emailing
- Manage and create report layouts using Microsoft Word 2013, SQL Server Report Builder (RDLC), or Visual Studio Community Edition
- Simplified payment reconciliation
- Encryption functions
- Report scheduling
- Data exchange framework for easier implementation

**Client enhancements**

Significant client enhancements include:

- Tablet client for multiple tablet types (iPad, Android, Windows)
- Enhanced web client
- Office 365 integration
- UI elements removal capability
- Simplified user interface option
- Enhanced Role Center Cues including color and image options

**Development tools**

Significant new development tools include:

- Enhanced commenting capability
- Non-default property values display bold font
- New Client Application Language (C/AL) commands and functions
An Introduction to NAV 2015

• New Development Environment commands
• New Report functions and status default to local status
• New Page field, action, and control properties
• New Table schema data synchronization options
• Windows PowerShell cmdlets and scripts include comparing and merging objects, caption management, and upgrading SQL server data when NAV table schema is changed

Other areas
Other new areas include:

• Major changes in Upgrade concepts and processes
• Deployment enhancements for Add-ins and .NET Framework
• PowerShell cmdlets for Exporting and Importing data selectively
• Licenses are now version specific
• RapidStart Services for implementation

A developer's overview of NAV 2015
From the point of view of a developer, NAV 2015 consists of about 4,000 potentially customizable off-the-shelf program objects plus the Integrated Development Environment (the C/SIDE development tools), which allow us to modify existing objects and create new ones.

The NAV 2015 system is an object-based system made up of the seven different object types available in NAV. Strictly speaking, NAV is not a full-featured object-oriented system. A full-featured object-oriented system would allow the definition and creation of new object types, while NAV only allows the creation and modification of instances of the seven predefined object types.
NAV object types
Let's start with basic definitions of the NAV 2015 object types:

- **Table**: Tables define the data structure and also contain the data records.
- **Page**: Pages are the way the data is formatted and displayed appropriately for each of the client types and user roles.
- **Report**: Reports provide display of the data to the user in hardcopy format, either on screen (preview mode) or via a printing device. Report objects can also update data in processes with or without data display.
- **Codeunit**: Codeunits are containers for code utilized by other objects. They are always structured in code segments called functions.
- **Query**: Queries support extracting data from one or more tables, making calculations, and outputting in the form of a new data structure. They can output data directly to charts, Excel, XML, and OData.
- **XMLport**: XMLports allow the importing and exporting of data to/from external files. The external file structure can be in XML or other file formats.
- **MenuSuite**: MenuSuites contain menu entries which refer to other types of objects. MenuSuites are different from other objects. Menus cannot contain any code or logic. MenuSuite entries display in the Departments page in the Navigation Pane in the Windows Client only. In the Web and Tablet clients, these are used to support Search functions.

The C/SIDE integrated development environment
NAV 2015 includes an extensive set of software development tools. The NAV development tools are accessed through C/SIDE which runs within the Development Environment client. This environment and its complement of tools are usually collectively referred to as C/SIDE. C/SIDE includes the C/AL compiler. All NAV programming uses C/AL. No NAV development can be done without using C/SIDE, but other tools are used to complement C/AL code (such as Visual Studio, .NET, COM controls, and OCX controls among others).
The C/SIDE is referred to as the **Object Designer** within NAV. It is accessed through a separate shortcut which is installed as part of a typical full system installation. When we open the Object Designer, we see the following screen:

![Object Designer tool icons](image)

**Object Designer tool icons**

When we open an object in the applicable Designer (Table Designer, Page Designer, and so on) for that object, we will see a set of tool icons at the top of the screen. The following table lists those icons and the object types to which they apply. Occasionally, an icon will appear when it is of no use.

<table>
<thead>
<tr>
<th>Type</th>
<th>C/AL Find</th>
<th>Properties</th>
<th>Field Menu</th>
<th>Symbol Table</th>
<th>C/AL Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>Table</td>
<td></td>
<td>✓</td>
<td>✓ but no response</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>Page</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Report</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Codeunit</td>
<td>✓</td>
<td>✓</td>
<td>✓ but empty</td>
<td>✓ Toggles debugger breakpoint</td>
<td>✓</td>
</tr>
<tr>
<td>Query</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>XMLPort</td>
<td>✓</td>
<td>✓</td>
<td>✓ but no response</td>
<td></td>
<td>✓</td>
</tr>
<tr>
<td>MenuSuite</td>
<td>✓ but of no use</td>
<td>No icon – Add+Enter</td>
<td></td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
The C/AL programming language

The language in which NAV is coded is C/AL. A small sample of C/AL code within the C/AL Editor is shown next:

C/AL syntax is similar to Pascal syntax. Code readability is always enhanced by careful programmer attention to structure, logical variable naming, process flow consistent with that of the code in the base product, and good documentation both inside and outside of the code.

Good software developer focuses on design before coding and on accomplishing design goals with a minimum of code. Dynamics NAV facilitates that approach. In 2012, a team made up of Microsoft and NAV community members began the NAV Design Patterns project. As defined on Wikipedia, "a design pattern is a general reusable solution to a commonly occurring problem". Links to the NAV Design Patterns project information are as follows:

- https://community.dynamics.com/nav/w/designpatterns/default.aspx

A primary goal of this project is to document patterns that exist within NAV. In addition, new best practice patterns have been suggested as ways to solve common issues we encounter during our customization efforts. Now, when working on enhancements of NAV, we will be aided by reference to the documentation of patterns within NAV. This allows us to spend more of our time designing a good solution using existing, proven functions (the documented patterns), and spending less time writing and debugging code. A good reference for NAV design and development using patterns can be found at https://www.packtpub.com/application-development/microsoft-dynamics-nav-2013-application-design.
To quote from the NAV 2015 Help:

"Reusing code makes developing applications both faster and easier. More importantly, if you organize your C/AL code as suggested, your applications will be less prone to errors. By centralizing the code, you will not unintentionally create inconsistencies by performing the same calculation in many places, for example, in several triggers that have the same table field as their source expression. If you have to change the code, you could either forget about some of these triggers or make a mistake when you modify one of them."

Much of our NAV development work is done by assembling references to previously defined objects and functions, adding new data structure where necessary. As the tools for NAV design and development provided by both Microsoft and the NAV community continue to mature, our development work becomes more oriented to design and less to coding. The end result is that we are more productive and cost effective on behalf of our customers. Everyone wins.

**NAV object and system elements**

Following are some important terms used in NAV:

- **License**: A data file supplied by Microsoft that allows a specific level of access to specific object number ranges. NAV licenses are very clever constructs which allow distribution of a complete system, all objects, modules, and features (including development) while constraining exactly what is accessible and how it can be accessed. Each license feature has its price, usually configured in groups of features. Microsoft partners have access to full development licenses to provide support and customization services for their clients. End-user firms can purchase licenses allowing them developer access to NAV. A Training License can also be generated which contains any desired set of features and expires after a specified period of time.
License limits
The NAV license limits access to the C/AL code within tables, pages, and codeunits differently than the C/AL code buried within reports or XMLports. The latter can be accessed with a lower level license (that is, less expensive). If a customer has license rights to the Report Designer, which many do, they can access C/AL code within Report and XMLport objects. But access to C/AL code in a table, page, or codeunit requires a more expensive license with Developer privileges. As a result, C/AL code within tables, pages, and codeunits is more secure than that within report and XMLport objects.

All licenses are now version-specific. From the Microsoft Dynamics NAV 2015 Licensing Guide: "since the release of Microsoft Dynamics NAV 2013 R2 CU10, license keys are version-specific. A Microsoft Dynamics NAV 2015 license key is required to activate Microsoft Dynamics NAV 2015 software and a Microsoft Dynamics NAV 2015 license key will not activate any other versions of the software."

• **Field**: An individual data item, defined either in a table or in the working storage (temporary storage) of an object.

• **Record**: A group of fields (data items) handled as a unit in many operations. Table data consists of rows (records) with columns (fields).

• **Control**: In MSDN, a control is defined as "a component that provides (or enables) UI capabilities".

• **Properties**: These are the attributes of the element such as an object, field, record, or control that define some aspect of its behavior or use. Example of property attributes include display captions, relationships, size, position, and whether editable or viewable.
• **Trigger**: Mechanisms that initiate (fire) an action when an event occurs and is communicated to the application object. A trigger in an object is either empty, contains code that is executed when the associated event fires the trigger, or contains only comments (in a few cases, this affects the behavior of the trigger). Each object type, data field, control, and so on, may have its own set of predefined triggers. The event trigger name begins with the word *On* such as OnInsert, OnOpenPage, or OnNextRecord. NAV triggers have similarities to those in SQL, but they are not the same (similarly named triggers may not even serve similar purposes). NAV triggers are locations within objects where a developer can place comments or C/AL code. When we view the C/AL code of an object in its Designer, we can see non-trigger code groups that resemble NAV event-based triggers.

  ° **Documentation**: It can contain comments only, no executable code. Every object type except MenuSuite has a single Documentation section at the beginning of the C/AL code.

  ° **Functions**: It can be defined by the developer. They are callable routines which can be accessed by other C/AL code from either inside or outside the object where the called function resides. Many functions are provided as part of the standard product. As developers, we may add our own custom functions as needed.

• **Object numbers and field numbers**: All objects of the same object type are assigned a number unique within the object type. All fields within an object are assigned a number unique within the object (that is, the same field number may be repeated within many objects whether referring to similar or different data).

> In this book, we will generally use comma notation for these numbers (fifty thousand is 50,000). In C/AL, no punctuation is used.

The object numbers from 1 (one) to 50,000 and in the 99,000,000 (99 million) range are reserved for use by NAV as part of the base product. Objects in these number ranges can be modified or deleted with a developer's license, but cannot be created. Field numbers in standard objects often start with 1 (one). Historically, object and field numbers from 50,000 to 99,999 are generally available to the rest of us for assignment as part of customizations developed in the field using a normal development license.
(At the time of this writing, the Developer Help says that object numbers below 50,000 can be used, at least for reports, but the authors’ testing doesn’t agree.) Field numbers from 90,000 to 99,999 should not be used for new fields added to standard tables as those numbers are sometimes used in training materials. Microsoft allocates ranges of object and field numbers to **Independent Software Vendor (ISV)** developers for their add-on enhancements. Some such objects (the 14,000,000 range in North America, other ranges for other geographic regions) can be accessed, modified, or deleted, but not created using a normal development license. Others (such as in the 37,000,000 range) can be executed, but not viewed or modified with a typical development license.

The following table summarizes object numbering practice:

<table>
<thead>
<tr>
<th>Object Number range</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 – 9,999</td>
<td>Base-application objects</td>
</tr>
<tr>
<td>10,000 – 49,999</td>
<td>Country-specific objects</td>
</tr>
<tr>
<td>50,000 – 99,999</td>
<td>Customer-specific objects</td>
</tr>
<tr>
<td>100,000 – 98,999,999</td>
<td>Partner-created objects</td>
</tr>
<tr>
<td>Above 98,999,999</td>
<td>Microsoft territory</td>
</tr>
</tbody>
</table>

- **Work Date**: This is a date controlled by the user operator. It is used as the default date for many transaction entries. The System Date is the date recognized by Windows. The Work Date, which can be adjusted at any time by the user, is specific to the workstation and can be set to any point in the future or the past. This is very convenient for procedures such as ending sales order entry for one calendar day at the end of the first shift, and then entering sales orders by the second shift dated to the next calendar day. There are settings to allow limiting the range of Work Dates allowed.
1. The Work Date can be set by clicking on arrowhead drop-down menu below the Microsoft Dynamics icon and selecting the Set Work Date... option.

2. Clicking on Set Work Date... in the drop-down options displays the Set Work Date screen. Or click on the date in the Status Bar at the bottom of the Role Tailored Client (RTC) window. In either case, we can enter a new Work Date:

**NAV functional terminology**

For various application functions, NAV uses terminology more similar to accounting than to traditional data processing. Some examples are listed as follows:

- **Journal**: A table of unposted transaction entries, each of which represents an event, an entity, or an action to be processed. There are General Journals for general accounting entries, Item Journals for changes to inventory, and so on.

- **Ledger**: A detailed history of posted transaction entries that have been processed. For example, General Ledger, Customer Ledger, Vendor Ledger, Item Ledger, and so on. Some Ledgers have subordinate detail ledgers, typically providing a greater level of quantity and/or value detail.
• **Posting:** The process by which entries in a Journal are validated, and then entered into one or more Ledgers.

• **Batch:** A group of one or more Journal entries posted at the same time.

• **Register:** An audit trail showing the history, by Entry No. ranges, of posted Journal Batches.

• **Document:** A formatted page such as an Invoice, a Purchase Order, or a Payment Check, typically one page for each primary transaction (a page may require display scrolling to be fully viewed).

### User interface

NAV 2015 UI is designed to be role oriented (also called role tailored). The term **role oriented** means tailoring the options available to fit the user's specific job tasks and responsibilities. If user access is via the Windows client, Web client, or Tablet client, the **Role Tailored Client (RTC)** will be employed. If the user access is via SharePoint or another client, the developer will have more responsibility to make sure the user experience is role tailored.

The first page that a user will meet is a Role Center Page. The Role Center Page provides the user with a view of work tasks to be done; it acts as the user's home page. The home Role Center Page should be tailored to the job duties of each user, so there will be a variety of Role Center Page formats for any installation.

Someone whose role is focused on Order Entry will likely see a different RTC home page than the user whose role focuses on Invoicing, even though both user roles are in what we generally think of as Sales and Receivables. The NAV 2015 RTC allows a great deal of flexibility for implementers, system administrators, managers, and individual users to configure and reconfigure screen layouts and the set of functions that are visible.
The key to properly designing and implementing any system, especially a role tailored system, is the quality of the User Profile analysis done as the first step in requirements analysis. User Profiles identify the day-to-day needs of each user's responsibilities relative to accomplishing the business' goals. Each user's tasks must be mapped to individual NAV functions or elements, identifying how those tasks will be supported by the system. A successful implementation requires the use of a proven methodology. It is very important that the upfront work be done and done well. Even the best programming cannot compensate for a bad definition of goals.

In our exercises, we will assume the upfront work has been well done and we will concentrate on addressing the requirements defined by our project team.
Hands-on development in NAV 2015

One of the best ways to learn a new set of tools, like those that make up a programming language and environment, is to experiment with them. We're going to have some fun doing that throughout this book. We're going to experiment where the cost of errors (otherwise known as learning) is small. Our development work will be a custom application of NAV 2015 for a relatively simple, but realistic application.

We're going to do our work using the Cronus demo database that is available with all NAV 2015 distributions and is installed by default when we install the NAV 2015 demo system. The simplest way to install the NAV 2015 demo is to locate all the components on a single workstation. A 64-bit system running Windows 7 Professional will suffice. Information about additional requirements is available in the MSDN library (https://msdn.microsoft.com/en-gb/default.aspx) under the heading System Requirements for Microsoft Dynamics NAV 2015. Other helpful information on installing NAV 2015 (the demo option is a good choice for our purposes) and addressing a variety of setup questions is available in the NAV 2015 area of the MSDN library. In fact, all the Help information for NAV 2015 is accessible in the MSDN library.

The Cronus database contains all the NAV objects and a small, but reasonably complete set of data populated in most of the system's functional applications areas. Our exercises will interface very slightly with the Cronus data, but not depend on any specific data values.

To follow along with all our exercises as a developer, you will need a developer license for the system with rights allowing the creation of objects in the 50,000 to 50,099 number range. This license should also allow at least read access to all the objects numbered below 50,000. If you don't have such a license, check with your Partner or your Microsoft sales representatives to see if they will provide a training license for your use.

The NAV 2015 development exercise scenario

Our business is a small radio station that features a variety of programs, news, music, listener call-in, and other program types. Our station call-letters are WDTU. Our broadcast materials come from several sources and in several formats: vinyl records, CDs, MP3s, and downloaded digital (usually MP3s). While our station has a large library, especially of recorded music, sometimes our program hosts (also called disc jockeys or DJs) want to share material from other sources. For that reason, we need to be able to easily add items to our play lists (the list of what is to be broadcast) and also have an easy-to-access method for our DJs to preview MP3 material.
Like any business, we have accounting and activity tracking requirements. Our income is from selling advertisements. We must pay royalties for music played, fees for purchased materials such as prepared text for news, sports, and weather information, and service charges for our streaming Internet broadcast service. As part of our licensed access to the public airwaves, a radio station is required to broadcast public service programming at no charge. Often that is in the form of Public Service Announcements (PSAs) such as encouraging traffic safety or reduction in tobacco use. Like all radio stations, we must plan what is to be broadcast (create schedules) and track what has been broadcast (such as ads, music, purchased programming, PSAs) by date and time. We bill our customers for the advertising, pay our vendors their fees and royalties, and report our public service data to the appropriate government agency.

Getting started with application design

Our design for our radio station will start with a Radio Show table, a Radio Show Card page, a Radio Show List page, and a simple Radio Show List Report. Along the way, we will review the basics of each NAV object type.

When we open the NAV Development Environment for the first time or to work on a different database, we must define what database should be opened. Navigate to File | Database | Open..., as shown in the following image:

Application tables

Table objects are the foundation of every NAV application. Tables contain data structure definitions, as well as properties that describe the behavior of the data, including data validations and constraints.
More business logic is required in complex applications than in simple data type validation, and NAV allows C/AL code to be put in the table to control insertion, modification, and deletion of records as well as logic on the field level. When the bulk of the business logic is coded at the table level, it is easier to develop, debug, support, modify, and even upgrade. Good design in NAV requires that as much of the business logic as possible reside in the tables. Having the business logic coded at the table level doesn't necessarily mean the code is resident in the table. NAV 2015 Help recommends the following guidelines for placing C/AL code:

In general, put the code in codeunits instead of on the object on which it operates. This promotes a clean design and provides the ability to reuse the code. It also helps enforce security. For example, typically, users do not have direct access to tables that contain sensitive data, such as the General Ledger Entry table, nor do they have permission to modify objects. If you put the code that operates on the general ledger in a codeunit, give the codeunit access to the table, and give the user permission to execute the codeunit, then you will not compromise the security of the table and the user will be able to access the table.

If you must put the code on an object instead of in a codeunit, then put the code as close as possible to the object on which it operates. For example, put code that modifies records in the triggers of the table fields.

Designing a simple table
Our primary master data table will be the Radio Show table. This table lists our inventory of shows available to be scheduled.

First, open the NAV Development Environment, click Tools | Object Designer and select Table. We can view or modify the design of existing master tables in NAV by highlighting the table (for example, Table 18 - Customer, or Table 27 - Item) and clicking on Design.

Each master table has a standard field for the primary key (a Code data type field of 20 characters called No.) and has standard information regarding the entity the master record represents (for example, Name, Address, City, and so on for the Customer table and Description, Base Unit of Measure, Unit Cost, and so on, for the Item table).
The Radio Show table will have the following field definitions (we may add more later):

<table>
<thead>
<tr>
<th>Field names</th>
<th>Definitions</th>
</tr>
</thead>
<tbody>
<tr>
<td>No.</td>
<td>20 character text (code)</td>
</tr>
<tr>
<td>Type</td>
<td>10 character text (code)</td>
</tr>
<tr>
<td>Name</td>
<td>50 character text</td>
</tr>
<tr>
<td>Run Time</td>
<td>Duration</td>
</tr>
<tr>
<td>Host No.</td>
<td>20 character text (code)</td>
</tr>
<tr>
<td>Host Name</td>
<td>50 character text</td>
</tr>
<tr>
<td>Average Listeners</td>
<td>Integer</td>
</tr>
<tr>
<td>Audience Share</td>
<td>Decimal</td>
</tr>
<tr>
<td>Advertising Revenue</td>
<td>Decimal</td>
</tr>
<tr>
<td>Royalty Cost</td>
<td>Decimal</td>
</tr>
</tbody>
</table>

In the preceding list, three of the fields are defined as Code fields, which are text fields that limit the alphanumeric characters to upper case values. Code fields are used throughout NAV for primary key values. They are used to reference or be referenced by other tables (foreign keys). The No. will be the unique identifier in our table. We will utilize a set of standard internal NAV functions to assign a user-defined No. Series range that will auto increment the value on table insertion and possibly allow for user entry (as long as it is unique in the table) based on a setup value. Host No. references the standard Resource table and the Type field will reference a custom table we will create to allow for flexible Type values.

We will have to design and define the reference properties at the field level in the Table Designer window, as well as compile them, before the validation will work. At this point, let's just get started with these field definitions and create the foundation for the Radio Show table.

**Creating a simple table**

To invoke the table designer, open the NAV 2015 Development Environment and the database in which we will be doing our development. In the Object Designer, click Table (in the left column of buttons) and click New (in the bottom row of buttons). Enter the first field number as 10 (the default is 1), and increment each remaining field number by 10 (the default is 1). Sometimes it is useful to leave large gaps (such as jumping from 80 to 200 or 500) when the next set of fields have a particular purpose not associated with the prior set of fields.
NAV 2015 Help says to not leave gaps in field numbers. Based on many years of experience, the authors disagree. Leaving numbering gaps will allow us to later add additional fields between the existing fields, if necessary. The result will be data structures that are (at least initially) easier to read and understand. Once a table is referenced by other objects or contains any data, the field numbers of the previously defined fields should not be changed.

In the Description column, we will put notes for the fields that need properties set later. The following image shows our new table definition in the Table Designer window:

Now we can close the table definition (navigate to File | Save or Ctrl + S or press ESC or close the window. The first two options are the explicit methods of saving our work). We will see a message reminding us to save our changes, as shown in the following screenshot:
Click Yes. We must now assign the object number (use 50000) and a unique name (it cannot duplicate the same first 30 characters of another table object in the database). We will name our table Radio Show based on the master record to be stored in the table.

In the preceding screenshot, note that the Compiled option is automatically checked and the Synchronize Schema option is set to Now – with validation, which are the defaults for NAV. Once we press OK and the object is successfully compiled, it is immediately ready to be executed within the application. If the object we were working on was not ready to be compiled without error, we could unselect the Compiled option in the Save As window.

Uncompiled objects will not be considered by C/SIDE when changes are made to other objects. Until we have compiled an object, it is a work in progress, not an operable routine. There is a Compiled flag on every object that gives its compilation status. Even when we have compiled an object, we have not confirmed that all is well. We may have made changes that affect other objects which reference the modified object. As a matter of good work habit, we should recompile all objects before we end work for the day.

The Synchronize Schema option choice determines how table changes will be applied to the table data in SQL Server. When the changes are validated, any changes that could be destructive to existing data will be detected and handled either according to a previously defined upgrade codeunit or by generating an error message. The Synchronize Schema option choices are shown in the following image:
Pages

Pages provide views of data or processes designed for on-screen display (or exposure as web services) and also allow for user data entry into the system. They act as containers for the action items (menu options).

There are several basic types of display/entry pages in NAV 2015, as listed next:

- List
- Card
- Document
- Journal/Worksheet
- List Plus
- Confirmation dialog
- Standard dialog

There are also page parts (they look and program like a page, but aren't intended to stand alone) as well as user interfaces that display like pages, but are not Page objects. The latter user interfaces are generated by various dialog functions. In addition, there are special Page types, such as Role Center pages and Navigate pages (for Wizards).

Standard elements of pages

A page consists of Page properties and triggers, controls, and control properties and triggers. Generally, Data controls are either labels displaying constant text or graphics, or containers that display data or other controls. Controls can also be buttons, action items, and page parts. While there are a few instances where we must include C/AL code within the page or page control triggers, it is good practice to minimize the amount of code embedded within pages. Any data-related C/AL code should be located in the table object rather than the page object.

See the Developer and IT ProHelp section Synchronizing Table Schemas for more details.
List pages

List pages display a simple list of any number of records in a single table. The Customer List page (with its associated FactBoxes) in the following screenshot shows a subset of the data for each customer displayed. Often the List pages/forms do not allow entry or editing of the data. Journal/Worksheet pages look like List pages, but are intended for data entry. Standard List pages are always displayed with the navigation pane on the left. The Customer List page is shown in the following screenshot:

Card pages

Card pages display one record at a time. These are generally used for the entry or display of individual table records. Examples of frequently accessed Card pages include Customer Card for customer data, Item Card for inventory items, and G/L Account Card for General Ledger accounts.

Card pages often have FastTabs (one FastTab consists of a group of controls with each tab focusing on a different set of related customer data). FastTabs can be expanded or collapsed dynamically, allowing the data visible at any time to be controlled by the user. Important data elements can be promoted to be visible even when a FastTab is collapsed.
Card pages for the master records display all the required data entry fields. If a field is set to **ShowMandatory** (a control property we will discuss in Chapter 4, Pages), a red asterisk will display until the field is filled. Typically, Card pages also display FactBoxes containing summary data about related activity. Thus, Cards can be used as the primary inquiry point for the master records. The following screenshot is a sample of a standard **Customer Card**:

![Customer Card](image)

**Document pages**

A document page looks like a Card page with one tab containing a List page. An example is the **Sales Order** page shown in the following screenshot. In this example, the first tab and the last four tabs are in Card page format showing sales order data fields that have a single occurrence on the page (in other words, they do not occur in a repeating column).
An Introduction to NAV 2015

The second tab from the top is in List page format (all fields are in repeating columns) showing the sales order line items. Sales order line items may include product to be shipped, special charges, comments, and other pertinent order details. The information to the right of the data entry area is related data and computations (FactBoxes) that have been retrieved and formatted. The top FactBox contains information about the ordering customer. The bottom FactBox contains information about the item on the currently highlighted sales line.
Journal/Worksheet pages

Journal and Worksheet pages look very much like List pages. They display a list of records in the body of the page. Many also have a section at the bottom that shows details about the selected line and/or totals for the displayed data. These pages may include a Filter pane and perhaps a FactBox. The biggest difference between the Journal/Worksheet pages and basic List pages is that Journal and Worksheet pages are designed to be used for data entry (though this may be a matter of personal or site preference). An example of the Requisition Worksheet page in Purchasing, is shown in the following screenshot. This Worksheet assists the user in determining and defining what purchases should be made.

Creating a List page

Now we will create a List page for the table we created earlier. A List page is the initial page that is displayed when a user accesses any data table. The NAV Development Environment has Wizards (object generation tools) to help create basic pages. After our Wizard work is done, we will spend additional time in the Page design tool to make the layout ready for presentation to users.
Our first List page will be the basis for viewing our Radio Show master records. From Object Designer, click Page, and then click New. The New Page screen will appear. Enter the name (Radio Show) or table object ID (50000) in the Table field. This is the table to which the page will be bound. We can add additional tables to the page object C/AL Global Variables after we close the wizard, as then we will be working in Page Designer. Choose the option Create a page using a wizard: and select List as shown in the following image. Click OK.

The next step in the wizard shows the fields available for the List page. We can add or remove any of the field columns by using the >, <, >>, and << buttons:
Add all the fields using >> and click Next >.

The next wizard step shows the Subforms, System FactBoxes, and Charts that are available to add to our page. (Subforms should properly be named Subpages. Such a change is being considered). We can add these later in Page Designer as needed. Click Finish to exit the wizard and enter Page Designer:
Click **Preview** to view the page with the default ribbon. Note that in the Preview mode, we cannot insert, modify, or delete any of the layout or enter data. The Preview page is not connected to the database data. We need to compile the page and run it to manipulate the data. In the following image of a Page Preview, some fields are out of sight at the right end:

![Page Preview](image1.png)

The Preview page is not connected to the database data. We need to compile the page and run it to manipulate the data. In the following image, some fields are out of sight at the right end:

![Page Preview](image2.png)
The availability of some capabilities and icons (such as OneNote) will depend on what software is installed on our development workstation. Close the preview of the List page and close the window or press \textit{ESC} to save. Number the page 50000 and name the object as Radio Show List, as shown in the following screenshot:

![Image of Save As dialog box with ID 50000, Name Radio Show List, and Compiled checked]

**Creating a Card page**

Next, let's create a Card page. The wizard process for a Card page is almost the same as for a List page, with one additional step. In Object Designer, with Pages selected, click \textbf{New} again. Enter the same table (Radio Show) and make sure the \textbf{Create a page using a wizard:} option is selected and \textbf{Card} is highlighted, as shown in the next screenshot:

![Image of New Page dialog box with Table Radio Show, Page Radio Show List, and Card highlighted]
The next step in the Wizard is specific to Card pages, which allows us to create **FastTabs**. These are the display tools that allow the user to expand or collapse the window sections for ease of viewing. For our Radio Show Card, we will divide our table fields into two sections, **General** (primary key, description, resource information, and duration) and **Statistics** (data about the show), as shown in the following screenshot:

![Card Page Wizard](image)

After defining the FastTab names, we must assign the data fields to the tabs on which they are to appear. We will populate the tabs based on the **FastTab Names** fields we assigned. We can select the fields from the **Available Fields** list and assign the order of appearance as we did in the List page wizard. Click the **Next >** button to proceed.
For the General FastTab, select the following fields: **No.**, **Type**, **Name**, **Run Time**, **Host Code**, and **Host Name**, as shown in the following screenshot:

Click the **Statistics** tab to populate the **Statistics** FastTab, select **Average Listeners**, **Audience Share**, **Advertising Revenue**, and **Royalty Cost**.
The last Card wizard step is to choose from the available Subforms (Subpages), System FactBoxes, and Charts. If we decide later that we want any of those, we will add them using Page Designer.

Click Finish to view the generated code in Page Designer.
Click the **Preview button** to show a view-only display of the card page.

Exit out of the Preview and Page Designer. Save the page as **ID 50001**, and **Name** as **Radio Show Card**. Refer to the following screenshot:

Later on we can add an action to the List page which will link to the Card page for inserting and editing radio show records and also add the List page to the Role Center page for our radio station user.
Creating some sample data

Even though we haven't added all the bells and whistles to our Radio Show table and pages, we can still use them to enter sample data. The Radio Show List page will be the easiest to use for this.

In Object Designer, with Pages selected, highlight Page 50000 – Radio Show List, and click Run. Then click the New icon on the ribbon. An empty line will open up where we can enter our sample data. Of course, since our table is very basic at this point, without any validation functionality, table references, function calls, and so on, we will have to be creative (and careful) and enter all the individual data fields accurately and completely on our own.

Enter the data shown in the following table so we can see what the page looks like when it contains data. Later on, after we add more capabilities to our table and pages, some fields will be validated, some will be either automatically entered or available on a lookup basis. But for now, simply key in each field value. If the data we key in now conflicts with the validations we create later (such as data in referenced tables), we may have to delete this test data and enter new test data later.

<table>
<thead>
<tr>
<th>No.</th>
<th>Type</th>
<th>Description</th>
<th>Resource Code</th>
<th>Resource Name</th>
<th>Run Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>RS001</td>
<td>TALK</td>
<td>CeCe and Friends</td>
<td>CECE</td>
<td>CeCe Grace</td>
<td>2 hours</td>
</tr>
<tr>
<td>RS002</td>
<td>MUSIC</td>
<td>Alec Rocks and Bops</td>
<td>ALEC</td>
<td>Alec Benito</td>
<td>2 hours</td>
</tr>
<tr>
<td>RS003</td>
<td>CALL-IN</td>
<td>Ask Cole!</td>
<td>COLE</td>
<td>Cole Henry</td>
<td>2 hours</td>
</tr>
<tr>
<td>RS004</td>
<td>CALL-IN</td>
<td>What do you think?</td>
<td>CLARK</td>
<td>Clark Ernest</td>
<td>1 hour</td>
</tr>
<tr>
<td>RS005</td>
<td>MUSIC</td>
<td>Quiet Times</td>
<td>FAY</td>
<td>Fay Mae</td>
<td>3 hours</td>
</tr>
<tr>
<td>RS006</td>
<td>NEWS</td>
<td>World News</td>
<td>GOLDIE</td>
<td>Goldie Nickles</td>
<td>1 hour</td>
</tr>
</tbody>
</table>
Creating a List Report

Open Object Designer, select Report and click New. Report Dataset Designer is empty when it displays, so we need to add a Data Source (table) to the first blank row. Type 50000 or Radio Show into the Data Source column.

To add multiple data fields from the table, we can use Field Menu which is accessed via the icon on the toolbar or the View | Field Menu option. Field Menu will show a list of all the fields in the Radio Show table:

Highlight the first six fields on the Field Menu. Then click on the next blank line in Report Dataset Designer:
A confirmation box will appear asking if we want to add the fields selected. Click Yes.

The fields will appear in Report Dataset Designer without having to type them in manually:

There are two options for RDLC Report Layout development tools: the current version of Visual Studio 2012 or 2014, or the free SQL Server Report Builder that matches the installed version of SQL Server. NAV defaults to using Visual Studio. But if the free SQL Server Report Builder is installed, it can be activated for NAV 2015 by accessing the Options... screen from the Tools menu option.
On the **Options** screen, set **Use Report Builder** to **Yes**.

![Options screen](image)

We can also now use the free Community version of Visual Studio, available at [https://www.visualstudio.com/en-us/downloads/visual-studio-2015-downloads-vs.aspx](https://www.visualstudio.com/en-us/downloads/visual-studio-2015-downloads-vs.aspx). If this link does not work, search the Web for **Visual Studio Community** or **Community Edition Visual Studio**. In this book, since we will use the free version of Community Visual Studio, we will not set **Use Report Builder** to **Yes**.

Click on **View** | **Layout** to proceed to the chosen report layout tool.
The RDLC Report Layout tool opens with a blank design surface and no visible dataset controls. Unlike Page Designer, there is no Report wizard to help with the layout of a new report. All the report layout design work must start from scratch with a blank design surface:

To show the dataset available from NAV, click View and select Report Data (the last item on the list). A new Report Data pane will show on the left of the Visual Studio layout window:
To create our simple list, we will insert a simple table object (a data region with fixed number of columns but variable number of rows) in the design surface. Right-click anywhere on the design surface and expand the Insert sub-menu to view the tools available on the report. Click the Table tool object, then use drag-and-drop to bring a control from the toolbox to the design surface, as shown in the following screenshot:

The table layout object defaults to three columns with a header row (repeated once) and a data row (repeated for each row of data retrieved from NAV).
An Introduction to NAV 2015

Drag and drop each of the six elements in **DataSet_Result** into columns in the table object. To add additional columns, right-click the table object header and select **Add Columns** (we could also drag-and-drop a field from the dataset to the table). The caption with the basic format of **Field Name Table Name** will default into the header row:

```
| No Radio Show | Type Radio | Name Radio | Run Time | Host Code | Host Name |
```

Let’s do a little cleanup in the header row by making the captions look like they do in standard NAV reports, by manually typing in the field names.

```
| No | Type | Name | Run Time | Host Code | Host Name |
```

We will save our work by clicking on **File | Save All** or clicking on **Ctrl + Shift + S** and then exit from Visual Studio (**File | Exit** or **Alt + F4**). Back in NAV Object Designer, we will exit out of the report or click on **File | Save**, then respond to two confirmation boxes. The first one asks if we want to save the report layout from Visual Studio. This allows us to load the RDLC report layout XML into the NAV database report object. Click **Yes**:
This is followed by the second confirmation screen. Enter 50000 for the ID, and Name the report Radio Show List. Click OK to save:

![Save As dialog box]

To view the report, make sure the new report object is selected, then click Run at the bottom of the Object Designer screen:

![Object Designer screen]
An instance of RTC will open with the Report Request Page showing. This is where the user can set filters, choose a sort sequence and choose the **Print..** options:

Click **Preview** to display the report on screen. The report will show our simple table layout with the fixed definition column captions showing exactly as we typed them.

All we've done so far is scratch the surface. But you should have a pretty good overview of the development process for NAV 2015.

You will be in especially good shape if you've been following along in your own system, doing a little experimenting along the way.
Other NAV object types
Let's finish up our introductory review of NAV's object types.

Codeunits
A codeunit is a container for chunks of C/AL code to be called from other objects. These chunks of code are called Functions. Functions can be called from any of the other NAV object types that can contain C/AL code. Codeunits can also be exposed (published) as web services. This allows the functions within a published codeunit to be invoked by external routines.

Codeunits are suited structurally to contain only functions. Even though functions could be placed in other object types, the other object types have superstructures that relate to their designed primary use as pages, reports, and so on.

Codeunits act only as a container for C/AL coded functions. They have no auxiliary functions, no method of direct user interaction, and no pre-defined processing. Even if we are creating only one or two functions and they are closely related to the primary activity of a particular object, if these functions are needed from both inside and outside of the report, the best practice is still to locate those functions in a Codeunit. For more guidance, see the NAV Design Pattern of the Week – the Hooks Pattern at [http://blogs.msdn.com/b/nav/archive/2014/03/16/nav-design-pattern-of-the-week-the-hooks-pattern.aspx](http://blogs.msdn.com/b/nav/archive/2014/03/16/nav-design-pattern-of-the-week-the-hooks-pattern.aspx).

There are several codeunits delivered as part of the standard NAV product which are actually function libraries. These codeunits consist totally of utility routines, generally organized on some functional basis (for example, associated with Dimensions or some aspect of Manufacturing or some aspect of Warehouse management). Many of these can be found by filtering the Codeunit Names on the strings "Management" and "Mgt" (the same could be said for some of the Tables with the string "Buffer" in their name). When we customize a system, we should create our own function library codeunits to consolidate our customizations and make software maintenance easier. Some developers create their own libraries of favorite special functions and include a "function library" codeunit in systems on which they work.

If a Codeunit is structured very simply and can operate in a stand-alone mode, it is feasible to test it in the same way one would test a Report or a Page. Highlight the Codeunit and click the Run button. The codeunit will run for a single cycle. However, most codeunits are more complex and must be tested by a calling routine.
Queries
Queries are objects whose purpose is to create extracted sets of data from the NAV database and do so very efficiently. NAV 2015 Queries translate directly into T-SQL query statements and run on the server side rather than on the service tier. A Query can extract data from a single table or multiple tables. In the process of extracting data, it can do different types of Joins (Inner Join, Outer Join, Cross Join), can filter, can calculate FlowFields (special NAV calculations which are discussed in detail in Chapter 3, Data Types and Fields), can sort, and can create sums and averages. Queries obey the NAV data structure business logic.

The output of a Query can be a CSV file (useful for Excel charts), an XML file (for charts or external applications), or an Odata file for a web service. Queries can be published for web service access in similar manner to Pages and Codeunits. The results of a Query can also be viewed by Pages (as described in Chapter 5, Queries and Reports) and Cues (as described in the Help Walkthrough: Creating a Cue based on a Normal Field and a Query), but are especially powerful when output to charts. With a little creativity, a Query can also be used to feed data to a report via use of a temporary table to hold the Query results.

MenuSuites
MenuSuites are the objects that are displayed in the navigation menus. They differ considerably from the other object types we have discussed earlier because they have a completely different structure and they are maintained differently. MenuSuite entries do not contain triggers. The only customization that can be done with them is to add, delete, or edit menu entries which are made up of a small set of properties.

In the RTC, the data in the MenuSuites object is presented in the Departments page.

XMLports
XMLports enable importing and exporting data. XMLports handle both XML structured data and other external text data formats. XML stands for eXtensible Markup Language which is the de facto standard for exchanging data between dissimilar systems. For example, XMLports could be used to communicate between our NAV ERP system and our accounting firm's financial analysis and tax preparation system.

XML is designed to be extensible, which means that we can create or extend the definition as long as we communicate the defined XML format to our correspondents. There is a standard set of syntax rules to which XML formats must conform. Much new software uses XML. For example, the new versions of Microsoft Office are quite XML friendly. All web services communications are in the form of an exchange of XML structured data.
The non-XML text data files handled by XMLports fall into two categories. One is known as comma separated value or comma delimited files (usually having a .csv file extension). Of course, the delimiters don't have to be commas. The other category is fixed format, in which the length and relative position of each field is pre-defined.

XMLports can contain C/AL logic for any type of appropriate data manipulation, either when importing or exporting. Functions such as editing, validating, combining, filtering, and so on, can be applied to the data as it passes through an XMLport.

**Development backups and documentation**

As with any system where we can do development work, careful attention to documentation and backing up of our work is very important. C/SIDE provides a variety of techniques for handling each of these tasks.

When we are working within Object Designer, we can back up individual objects of any type or groups of objects by exporting them. These exported object files can be imported in total, selectively in groups, or individually, to recover the original version of one or more objects.

NAV 2015 introduces Windows PowerShell cmdlets that support backing up data to the NAVData files. Complementary cmdlets support getting information or selectively retrieving data from previously created NAVData files. Although these tools promise to be very handy for repetitive development testing, they are challenging (or worse) to use in an environment of changing table or field definitions.

When objects are exported to text files, we can use a standard text editor to read or even change them. If, for example, we want to change all the instances of the field name **Customer** to **Patient**, we might export all the objects to text and execute a mass *Find and Replace*. Making such code changes in a text copy of an object is subject to a high probability of error, as we won't have any of the many safety features of the C/SIDE environment to limit what we can do.

Internal documentation (that is, inside C/SIDE) of object changes can be done in three areas. First is the **Object Version List**, a field attached to every object, visible in the Object Designer screen. Whenever a change (or set of changes) is made in an object, a notation should be added to the Version List.

The second area for documentation is the **Documentation trigger** that appears in every object type except MenuSuites. The Documentation trigger is at the top of the object and is the recommended location for noting a relatively complete description of any changes that have been made to the object. Such descriptions should include a brief description of the purpose of the change as well as technical information.
The third area we can place documentation at is inline with modified C/ AL code. Individual comment lines can be created by starting the line with double forward slashes //. Whole sections of comments (or commented out code) can be created by starting and ending the section with a pair of curly braces { }. Depending on the type of object and the nature of the specific changes, we should generally annotate each change inline wherever the code is touched, so all the changes can be easily identified by the next developer.

In short, when doing development in NAV C/SIDE, everything applies of what we have learned earlier about good documentation practices. This holds true whether the development is new work or modification of existing logic.

Summary
In this chapter, we covered some basic definitions of terms related to NAV and C/SIDE. We followed it with the introduction of the seven NAV objects types (Tables, Pages, Reports, Codeunits, Queries, MenuSuites, and XMLports). We introduced Table, Page, and Report creation through review and hands-on use beginning a NAV application for the WTDU Radio Show programming management. Finally, we looked briefly at the tools that we use to integrate with external entities and discussed how different types of backups and documentation are handled in C/SIDE. Now that we have covered the basics, we will dive into the detail of the primary object types in the next few chapters.

In Chapter 2, Tables, we will focus on Tables, the foundation of a NAV system.

Review questions
Q 1. An ERP system such as NAV 2015 includes a number of functional areas. Which of the following are part of NAV 2015? Choose four.

a. Manufacturing  
b. Order Processing  
c. Planning  
d. Computer Aided Design (CAD)  
e. General Accounting
Q 2. New functionality in NAV 2015 includes (choose three):

   a. Tablet client
   b. Multi-language
   c. Document emailing
   d. Spell checker
   e. Mandatory fields

Q 3. NAV 2015 development is done in the C/SIDE IDE and Visual Studio. True or False?

Q 4. Match the following table types and descriptions for NAV.

<table>
<thead>
<tr>
<th>Table types</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Journals</td>
<td>a Audit trail</td>
</tr>
<tr>
<td>2 Ledgers</td>
<td>b Validation process</td>
</tr>
<tr>
<td>3 Register</td>
<td>c Invoice</td>
</tr>
<tr>
<td>4 Document</td>
<td>d Transaction entries</td>
</tr>
<tr>
<td>5 Posting</td>
<td>e History</td>
</tr>
</tbody>
</table>

Q 5. iPads can be used with NAV 2015 to display the Role Tailored Client. True or False?

Q 6. Which of the following describe NAV 2015? Choose two.

   a. Customizable
   b. Includes a Storefront module
   c. Object based
   d. C# IDE
   e. Object oriented

Q 7. What are the seven NAV 2015 object types?

Q 8. All NAV objects except XMLports can contain C/AL code. True or False?

Q 9. NAV 2015 includes support for publishing objects as Web Services. True or False?
Q 10. What is the "home page" for a NAV 2015 user called? Choose one.
   a. Role Home
   b. Home Center
   c. Main Page
   d. Role Center

Q 11. Page Previews from the Development environment can be used for data entry and maintenance. True or False?

Q 12. For what work is Visual Studio used in NAV 2015? Choose one.
   a. Report data definition
   b. Report layout
   c. Role Center design
   d. Query processing

Q 13. Codeunits are the only NAV 2015 objects that can contain functions. True or False?

Q 14. Query output can be used as a Data Item for Reports. True or False?

Q 15. C/AL and C/SIDE are required for NAV 2015 development. True or False?

Q 16. What object number range is available for assignment to customer-specific objects? Choose two.
   a. 20-500
   b. 50000 – 60000
   c. 150000 – 200000
   d. 50000 – 99999
   e. 10000 – 100000

Q 17. XMLports can only process XML-formatted data. True or False?

Q 18. The Work Date can only be changed by the System Administrator. True or False?
Q 19. A Design Pattern is which of the following? Choose two.
   a. Reusable code
   b. Stripes and plaid together
   c. A proven way to solve a common problem
   d. User Interface guidelines

Q 20. NAV 2015 Reports are often generated automatically through the use of a wizard. True or False?
Where to buy this book

You can buy Programming Microsoft Dynamics™ NAV 2015 from the Packt Publishing website.

Alternatively, you can buy the book from Amazon, BN.com, Computer Manuals and most internet book retailers.

Click here for ordering and shipping details.