



POWER OF SIMPLICITY

Getting Started With TDL

The information contained in this document is current as of the date of publication and subject to change. Because Tally must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Tally, and Tally cannot guarantee the accuracy of any information presented after the date of publication. The information provided herein is general, not according to individual circumstances, and is not intended to substitute for informed professional advice.

This document is for informational purposes only. TALLY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT AND SHALL NOT BE LIABLE FOR LOSS OR DAMAGE OF WHATEVER NATURE, ARISING OUT OF, OR IN CONNECTION WITH THE USE OF OR INABILITY TO USE THE CONTENT OF THIS PUBLICATION, AND/OR ANY CONDUCT UNDERTAKEN BY PLACING RELIANCE ON THE CONTENTS OF THIS PUBLICATION.

Complying with all applicable copyright and other intellectual property laws is the responsibility of the user. All rights including copyrights, rights of translation, etc., are vested exclusively with TALLY SOLUTIONS PRIVATE LIMITED. No part of this document may be reproduced, translated, revised, stored in, or introduced into a retrieval system, or transmitted in any form, by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Tally Solutions Pvt. Ltd.

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

© 2009 Tally Solutions Pvt. Ltd. All rights reserved.

Tally.Developer 9 is either registered trademarks or trademarks of Tally Solutions Pvt. Ltd. in India and/or other countries. All other trademarks are properties of their respective owners.

Version: Getting Started With TDL/1.0/July 2009

Contents

Lesson 1: Introduction

| | |
|--|---|
| 1.1 Introduction to Tally.ERP 9 | 1 |
| 1.2 About the Product Tally.Developer 9 | 2 |
| 1.2.1 Tally Developer Installation - An Introduction | 2 |
| 1.2.2 Pre-Installation - System Requirements | 2 |
| 1.2.3 Steps for Installing Tally.Developer 9 | 3 |
| 1.2.4 Uninstalling Tally.Developer 9 | 7 |
| 1.3 Tally Definition Language | 7 |
| 1.3.1 What is TDL? | 7 |
| 1.3.2 Configuration in Tally.ERP 9 to enable TDL | 8 |

Lesson 2: Getting Started With TDL

| | |
|--|----|
| 2.1 Creating your first TDL Program | 11 |
| 2.1.1 Steps for creating TDL program using Tally.Developer 9 | 11 |
| 2.2 Understanding the Various Components of TDL | 13 |
| 2.2.1 Definitions | 13 |
| 2.2.2 Attributes & Modifiers | 16 |
| 2.2.3 Data Types in TDL | 17 |
| 2.2.4 Operators in TDL | 18 |
| 2.2.5 Special Symbols | 20 |
| 2.2.6 Actions in TDL | 20 |
| 2.2.7 Functions | 21 |
| 2.3 Variables in TDL | 22 |
| 2.3.1 Types of Variable | 22 |
| 2.3.2 Attributes of Variable Definition | 23 |

Lesson 3: Reports in TDL

| | |
|--|----|
| 3.1 Display Reports | 26 |
| 3.1.1 Understanding Dimensions & Formatting | 27 |
| 3.1.2 Tally Data Structure- Objects, Methods and Collections | 31 |
| 3.2 Edit Reports | 37 |
| 3.3 Designing Reports using Existing Data | 39 |
| 3.3.1 Simple Trial Balance Report | 39 |
| 3.3.2 Tabular Reports | 41 |
| 3.3.3 Hierarchical Report (Drill down Report) | 43 |

Lesson 4: Customisation – An Insight

| | |
|---|----|
| 4.1 Customising Default Screens | 47 |
| 4.1.1 User Defined Fields | 47 |
| 4.1.2 Voucher Customization | 55 |
| 4.1.3 Customisation - Case studies | 56 |
| 4.2 Invoice Customisation | 60 |
| 4.2.1 Invoice Customization - User defined format | 60 |
| 4.2.2 Invoice Customization - Modifications to default format | 64 |

Lesson 1: Introduction

Lesson Objectives

On completion of this lesson, you will be able to

- Know the products Tally.ERP 9 and Tally.Developer 9
- Know the installation and uninstallation procedure for Tally.Developer 9
- Understand the basic features of Tally Definition Language

1.1 Introduction to Tally.ERP 9

For over the years, Tally's drive has been to constantly develop cutting edge technology that has practical relevance to businesses. New features, new services, new technologies and the power of Tally's simplicity have made Tally the most used business solution in India and earned us worldwide acceptance.

Tally.ERP 9 - It's Fast, Powerful, Scalable... And very reliable!

Tally.ERP 9 is the world's fastest and most powerful concurrent Multi-lingual business Accounting and Inventory Management Software. Tally.ERP 9, designed exclusively to meet the needs of small and medium businesses, is a fully integrated, affordable and highly reliable software. Tally.ERP 9 is easy to buy, quick to install, and easy to learn and use.

Tally.ERP 9 is designed to automate and integrate all your business operations, such as sales, finance, purchasing, inventory, and manufacturing. With Tally.ERP 9, accurate, up-to-date business information is literally at your fingertips anywhere. The powerful new features and blazing speed and power of Tally.ERP 9 combine with enhanced MIS, Multi-lingual, Data Synchronization and Remote capabilities help you simplify all your business processes easily and cost-effectively.

Tally Definition Language (TDL) is a proprietary language of Tally. TDL enables customisation of Tally.ERP 9 to incorporate additional functionalities. The product Tally.Developer 9 is designed to be an integrated development environment with intelligence for development in the TDL language.

1.2 About the Product Tally.Developer 9

Tally.Developer 9 is a comprehensive Tally Definition Language (TDL) development environment designed specifically for TDL programmers. TD helps to generate customised Tally applications quickly. Tally.Developer 9 makes coding and development of TDL programs much easier compared to any other generic text editor.

Tally.Developer 9 has the ability to compile program source codes as a Tally Compiled Programs i.e. a .tcp file. A .tcp file is a compiled TDL project which can be executed from within Tally.ERP 9 and it is only readable by Tally. A TCP program file contains product information provided by developers at the time of project creation. Tally.Developer 9 provides users with the capability to code efficiently in TDL and to deliver the codes in compiled formats.

The following are the some features of Tally.Developer 9:

- ❑ Syntax checking and highlighting
- ❑ Hyperlink for the existing definitions
- ❑ Browsing for Default TDL codes
- ❑ TDL language browsers - Project browser, Definition Browser, Schema browser etc.
- ❑ Tally Developer Tools like Encrypted TDL migration, Decompilation and Tally connector

1.2.1 Tally Developer Installation - An Introduction

On successful installation of Tally.Developer 9, a shortcut is placed on the desktop, a folder titled Tally.Developer 9 is created in the selected drive and all the files required to run Tally.Developer 9 are stored in this default folder. Alternatively, one can also specify another path, if required.

1.2.2 Pre-Installation - System Requirements

Before installing Tally.Developer 9, please ensure that basic operational rights on the system i.e., read, write access is available. The Hardware requirements and Operating system required for a Client-Server and a standalone computer are as listed below:

Minimum Hardware requirements for Tally Developer 9

| | |
|-----------------------------|--|
| Processor | Intel Pentium IV or above and Equivalent |
| Memory | 256 MB RAM (Recommended 512 MB or more) |
| Free Hard Disk Space | 40 MB Minimum |
| Monitor Resolutions | 800 x 600 (Recommended 1024 x 768 or Higher) |

Operating Systems Supported

| | |
|--------------------|---|
| Single User | Microsoft Windows 98/NT/2000/2003/2008/XP/Vista |
| Multi-User | License Server can be installed only on Windows NT/2000/2003/XP/Vista |

1.2.3 Steps for Installing Tally.Developer 9

You can install **Tally.Developer 9** using the following method:

1. Double click the **install.exe** icon

Follow the instructions displayed on your screen to proceed with the installation of **Tally.Developer 9**.

2. The Tally.Developer 9 Setup Wizard screen is displayed as shown.

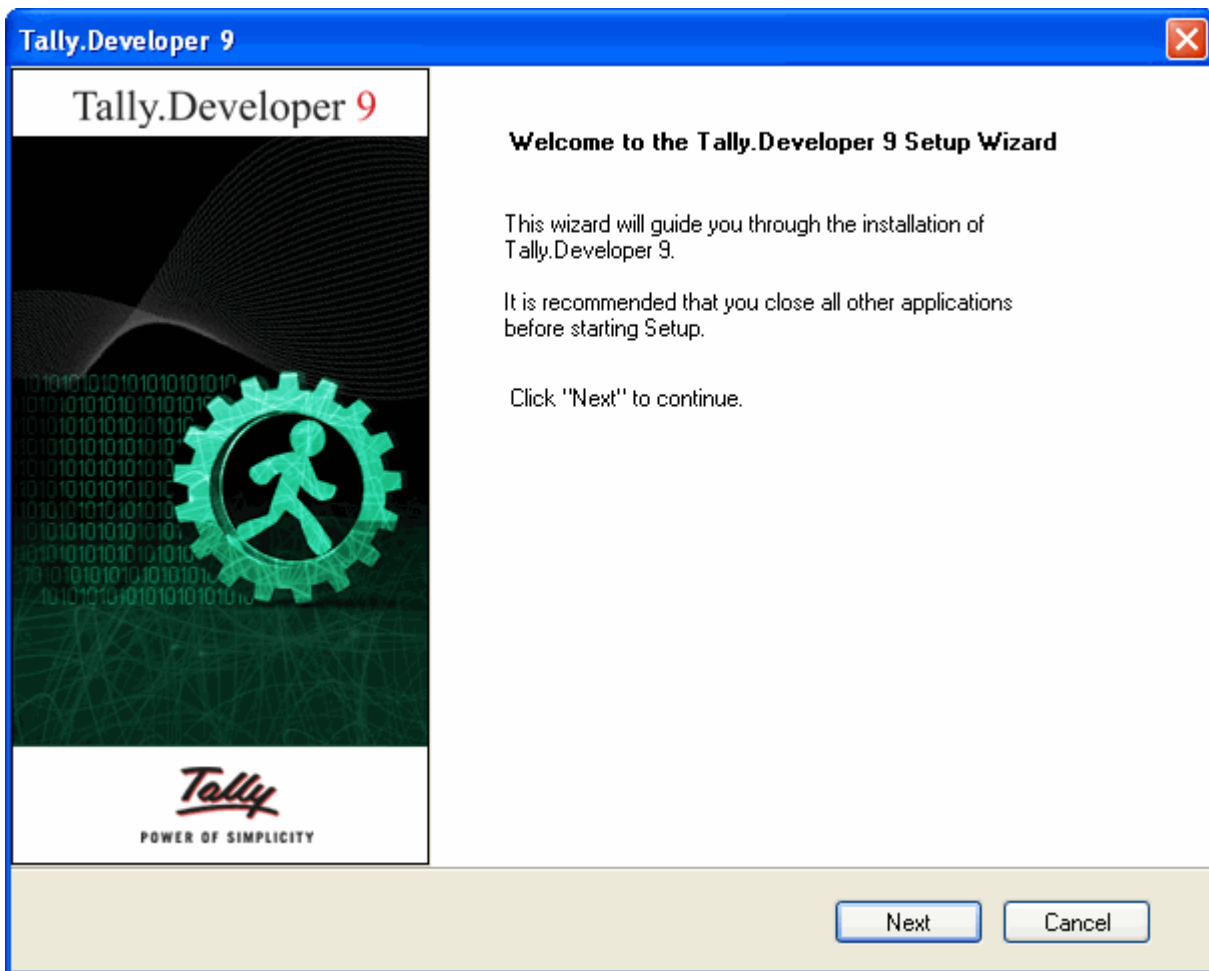


Figure 1.1 Tally.Developer 9 Setup Wizard

3. Click **Next** to proceed with the installation. On clicking Next, the following screen appears:

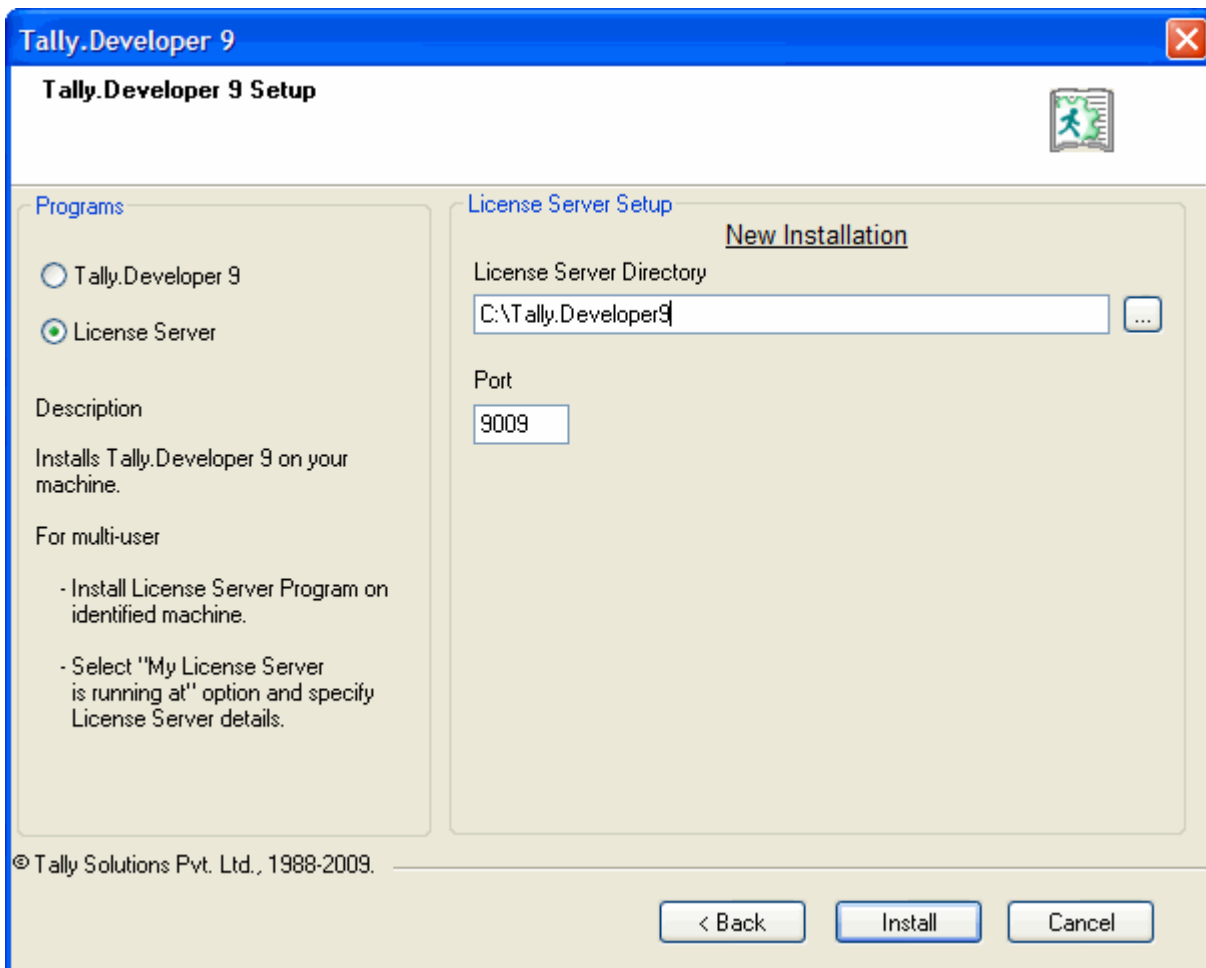


Figure 1.2 Tally.Developer 9 Setup Screen

- In the **Programs** section, ensure that **Tally.Developer 9/License Server** is checked. (Here we are selecting License Server)
- **New Installation** appears as Title for fresh installation or installation in a new folder. If Tally.Developer 9 is already installed on the system, the header is displayed as **Upgradation**.
- In the **License Server Directory/Application Directory**, you can either
 - Accept the directory that appears by default i.e., C:\Tally.Developer9 OR
 - Click the browse button and choose an existing directory OR
 - Type in the Path as required

If the specified path is not found, Installer creates a New Folder as specified by you

4. After specifying the required path, click **Install**
5. The **Setup status** screen appears as shown.

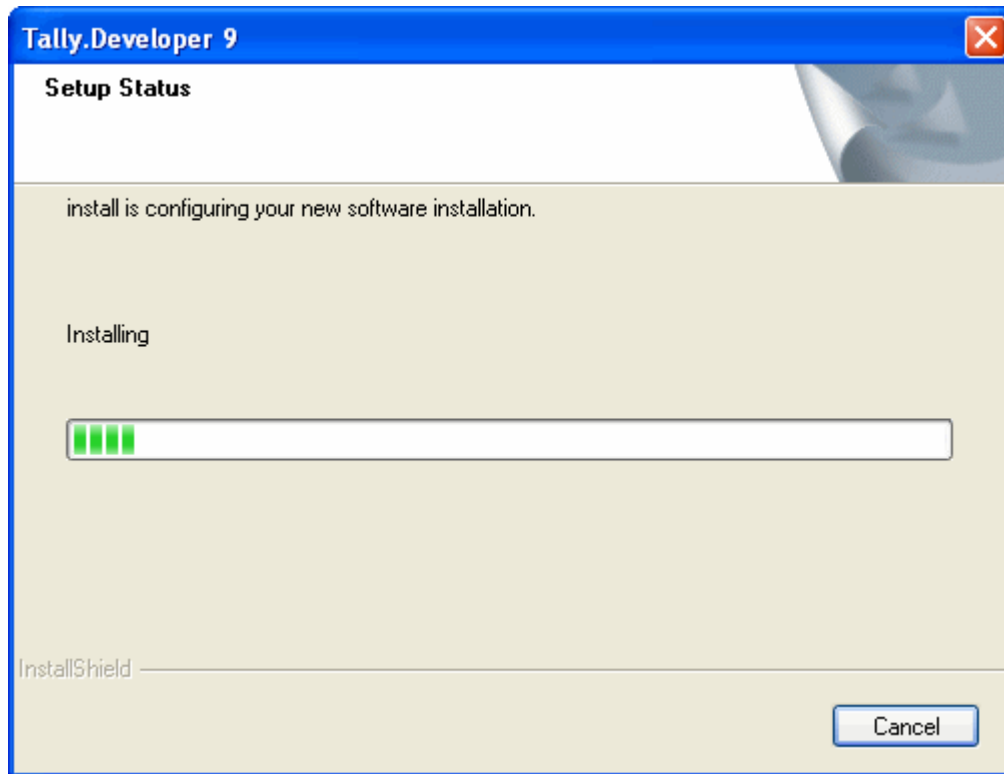


Figure 1.3 Tally.Developer 9 Setup Status

6. In the installer screen, click **Finish** to complete the setup.

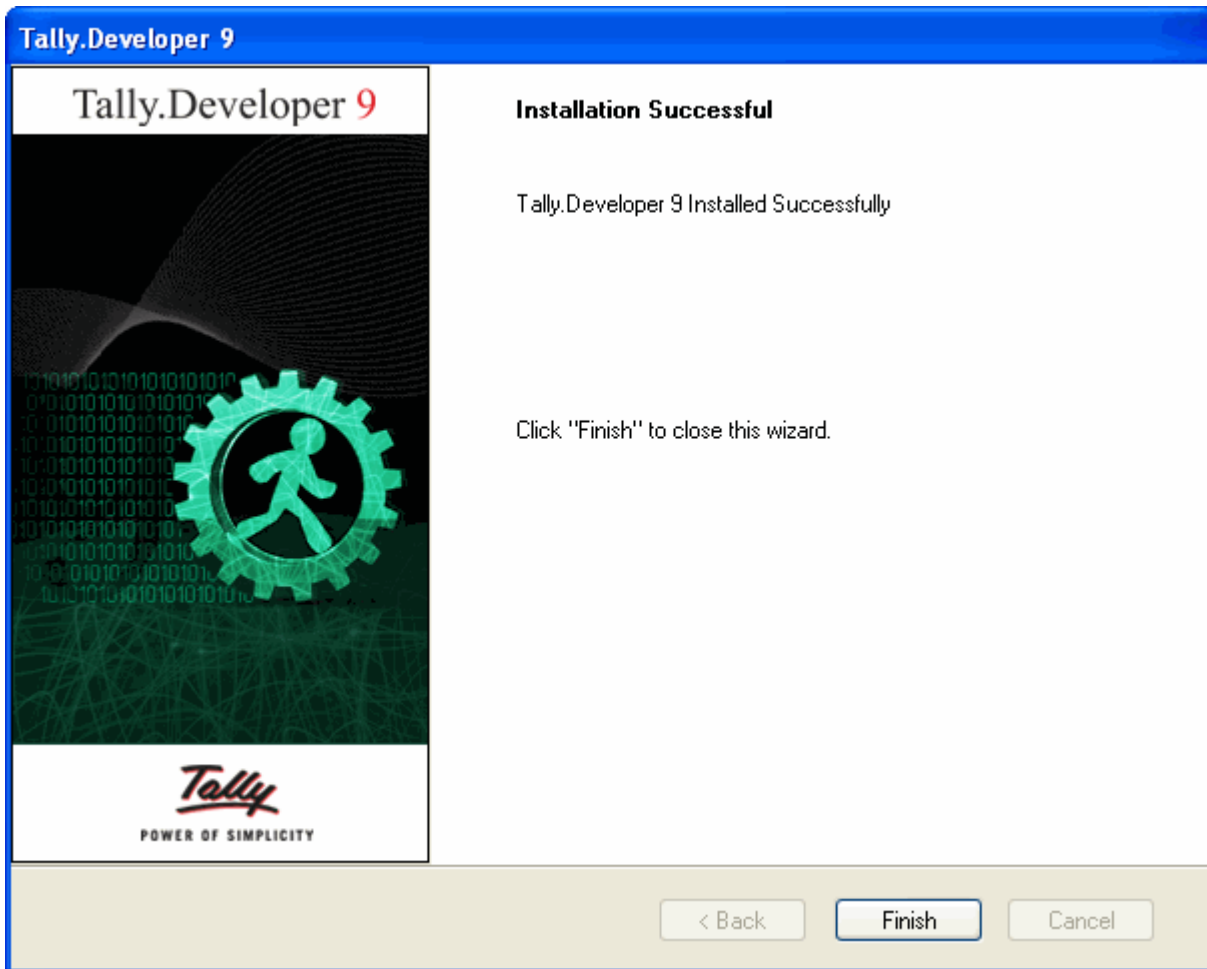


Figure 1.4 Installation Successful Screen

An icon named **Tally.Developer 9** is found on the Desktop on completion. Double Click the same to start working with Tally.Developer 9.



On the successful installation of Tally.Developer 9 following documents are available in the Tally.Developer 9 folder: Extending Tally.ERP9 using TDL, TDL Enhancements for Tally.ERP 9 and corresponding sample codes are available in folder Samples.

1.2.4 Uninstalling Tally.Developer 9

To uninstall Tally.Developer 9 click the icon Uninstall from the Tally.Developer 9 folder. Select the option for uninstall and click ok.

1.3 Tally Definition Language

A programming language is usually a series of commands, used as instructions, to a computer, specifying the task to be accomplished. Programming languages can be of two different types, namely, Procedural Language and Definition Language. Procedural language provides commands that can be used to define the task to be performed. The commands are executed in the order specified by the user. C is an example of procedural language.

Definition language provides users with 'Definitions' that can be used to specify the task to be performed. The user can specify the task to be performed, but has no control over the sequence of events that occur while performing the specified task. The sequence of events is implicit to the language and cannot be changed by the user.

The most powerful and important technical capability which is available in Tally.ERP 9 is the **Tally Definition Language (TDL)**. This is a rapid business application development language using which Tally.ERP 9 itself has been developed. Almost anything in Tally.ERP 9 can be altered, customised, extended using this language.

Tally Definition Language (TDL) enables customisation of Tally to incorporate additional functionalities. TDL is platform independent, which means TDL programs remain the same, irrespective of the Operating System/ Network Environment one uses. TDL also allows the sharing of data across different platforms.

1.3.1 What is TDL?

TDL is a proprietary language used in Tally. It is very specific to Tally and not suitable for any other environment. It is a language based on Definitions, which is non-procedural by concepts. As a definition language, TDL allows the user to define the required tasks, but it has a set of implicit events that occur by default over which the user has no control.

The following image gives the structure of Tally - Application Development Platform. There are three different layers: Hardware Layer, Platform Layer and TDL Layer

Any user of Tally.ERP 9 can learn TDL and develop extensions for the product. The entire source code of the product is available as part of the Tally Development Environment i.e. with our product Tally.Developer9.

Tally Definition Language provides a development platform for the TDL programmer. TDL provides flexibility & power to extend the default capabilities of Tally and integrate with external applications

The capabilities of TDL language are:

- ▣ **Rapid Development:** TDL definition is possible to reuse the existing definitions and deploy them. It is possible to develop complex reports with a short span of time.
- ▣ **Multiple Output Capability:** TDL language can be used to send the output to multiple output devices and formats

- **Data Management Capability:** Everything in TDL is an object. The data stored and retrieved as objects. By using TDL the user can create a new field and store a value into it which can be persisted in the Tally.ERP 9 database
- **Integration Capability:** The Tally.ERP 9 platform has a built in capability of integrating data with other data sources: Different data sources possible in Tally.ERP 9 are XML, ODBC, DLL, EXCEL etc.

Using the capabilities listed above, we can achieve the following

- Invoice Printing, Payment Advice Printing, Voucher Printing, etc. in user desired pre-pre-printed or plain formats
- Various Columnar reports like Batchwise Itemwise Reports, Itemwise Partywise Outward and Inward Movement Analysis
- Various security related controls like Voucher Type wise Entry Control, control the table of selection based on users like Sales persons can view only Debtors Ledgers, etc.
- Customization of Synchronization i.e., One way sync, Masters Only, Voucher Type wise Sync between various branches and HO
- Creation of multiple Approval Levels
- Labels and Bar Code Printing
- Auto creation of Masters/ Transactions as required

As discussed earlier, **TDL is a definition language** which provides the users with 'Definitions' that can be specified the task to be performed. And it has no control over the sequence of events, which is implicit to the language, that occur while performing the specified task

TDL is a non procedural language because the TDL programmer cannot control the sequence of events. The platform provides a set of functions for the TDL programmer

TDL is an Action Driven Language, the programmer can only control as to what happens when a particular event takes place. While interaction, the user can select any sequence of action. Based on his/her action a particular code gets executed.

TDL is a Rich language, which refers to a list of functions, attributes, and actions etc. which are provided by the platform.

1.3.2 Configuration in Tally.ERP 9 to enable TDL

To enable TDL in Tally.ERP 9:

1. Run Tally.ERP 9 instance
2. Click **F12:Configure** button to display the Configuration menu
3. Select the menu item **TDL Configuration**
4. Press **F4** or click **Local TDLs**
5. Set the value **Yes** for the option '**Load TDLs on Start up**'
6. Specify the **<Path\filename>** with extension in **List of TDLs to preload on Tally Start-up** field

The following image shows the TDL configuration screen. Here the user can attach either .tcp file or .txt file.

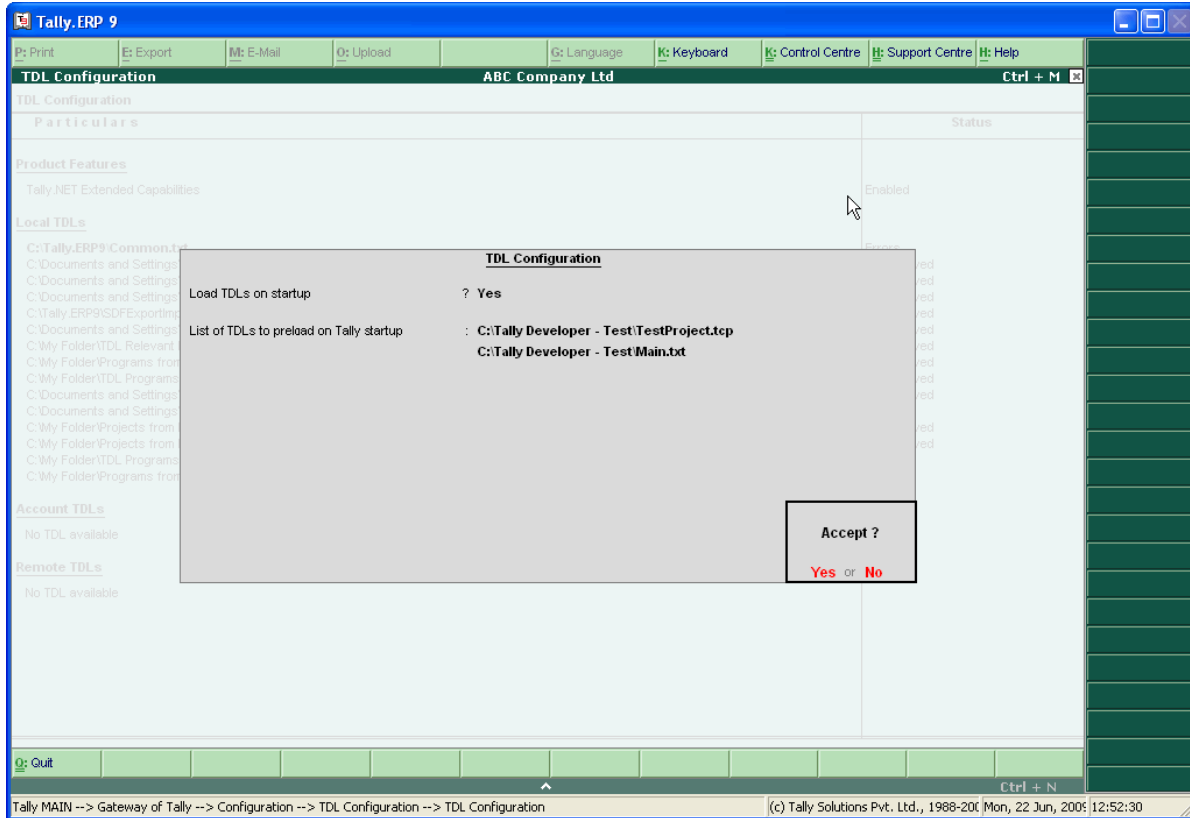


Figure 1.5 TDL Configuration in Tally.ERP 9

The following screen shows whether the attached TDLs are active or not allowed. The user can comment the file using '#' key.

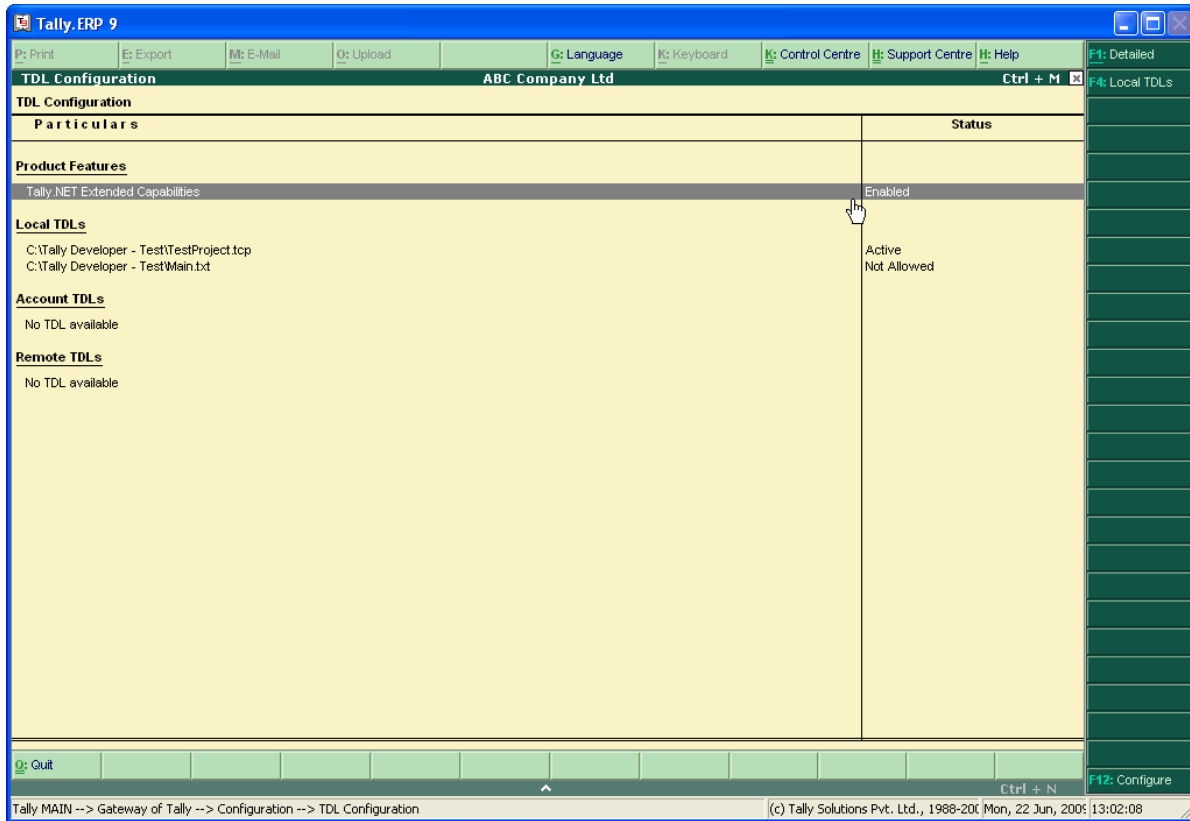


Figure 1.6 TDL Configuration

Lesson 2: Getting Started With TDL

Lesson Objectives

On completion of this lesson, you will be able to

- ❑ Understand the basic structure of a TDL program
- ❑ Understand the various components of a TDL program
- ❑ Understand the usage of variables in TDL

2.1 Creating your first TDL Program

2.1.1 Steps for creating TDL program using Tally.Developer 9

- ❑ Create a project or file in Tally.Developer 9
- ❑ Add Program Files to the project using the project properties option
- ❑ To add files to the project
- ❑ Using the option New File from the File menu the user can create .txt or .tdl file. The user can enter the TDL statement in editor window.
- ❑ Compile, Run and Test your program in Tally.ERP 9
- ❑ Attach .tcp file in Tally configuration screen

As we discussed, TDL is a language based on definitions. When we start Tally.ERP 9 the Interfaces which are visible on the screen are Menu, Report, Button and Table.

A Report and Menu can exist independently. A Menu is created by adding items to it while a Report is created using Form, Part, Line and Field. These are the definitions which cannot exist without a Report. TDL operates through the concept of an action which is to be performed and Definition on which the action is performed. The Report is invoked based on the action.

A basic TDL program to create a Report contains the definition Report, Form, Part, Line and Field and an action to execute the Report. A Report can have more than one Form, Part, Line and Field definitions but at least one has to be there. The hierarchy of these definitions is as follows:

- Report uses a Form
- Form uses a Part
- Part uses a Line
- Line uses a Field
- A Field is where the contents are displayed or entered. The Report is called either from a Menu or from a Key event.

Consider the following example to understand the creation of a basic TDL program. This TDL program will add a new menu item, First TDL, in the Gateway of Tally menu.

```
[#Menu: Gateway of Tally]
    Item: First TDL : Display : First TDL Report

[Report : First TDL Report]
    Form : First TDL Form

[Form : First TDL Form]
    Parts : First TDL Part

[Part : First TDL Part]
    Lines : First TDL Line

[Line : First TDL Line]
    Fields : First TDL Field

[Field : First TDL Field]
    Set as : "Welcome to the world of TDL"
```

The above code will add a new menu item in the **Gateway of Tally** menu and it displays the text "Welcome to the world of TDL". When the Menu Item is selected the report, First TDL Report is displayed. The report is in display mode as the action '**Display**' is specified while adding the menu item '**First TDL**'. The user input is not accepted in this report.

Consider the following image for better understanding of a basic TDL program.

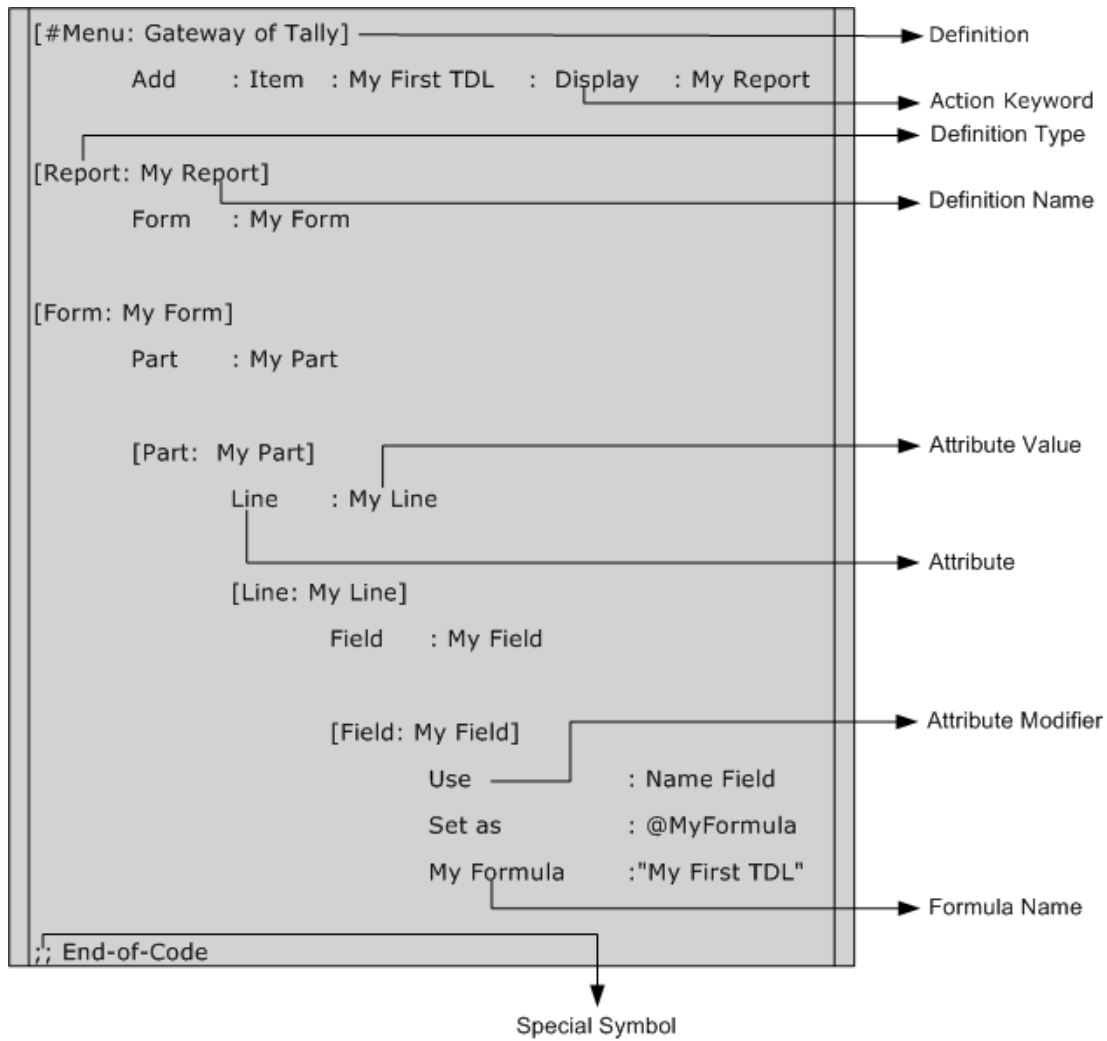


Figure 2.1 TDL Program - At a Glance

2.2 Understanding the Various Components of TDL

TDL consists of Definitions, Attributes & Modifiers, Datatypes, Operators, Special Symbols, Actions and Functions.

2.2.1 Definitions

TDL works on named definitions. The biggest advantage of working with TDL is its reusability of definitions. All the definitions are reusable by themselves and can be a part of other definitions.

Example:

```
[Part: FirstPart]
```

In the example mentioned above,

Part is the name of predefined definition type available in the platform. Menu, Report, Form, Part, Line, Field etc are the examples of definition names.

All definitions start with an open square bracket and end with a closed bracket

FirstPart is the user defined name which the user provides to instantiate the definition i.e. whenever a definition is created, a new object of a particular definition type comes into existence.

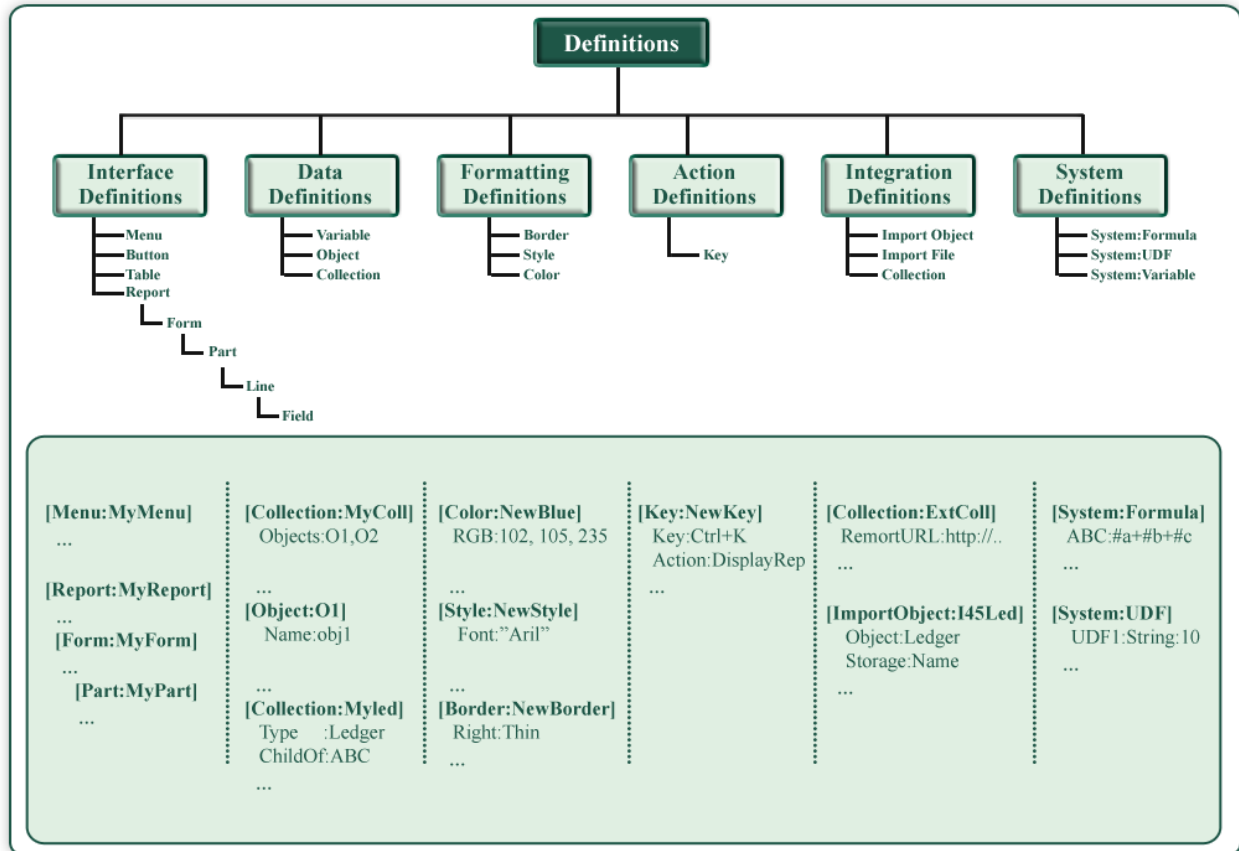


Figure 2.2 Definitions

Types of Definition

The various definitions in TDL are categorized as follows:

- ❑ Interface Definitions – Menu, Report, Form, Part, Line, Fields, Button, Table
- ❑ Data Definitions – Object, Variable, Collection
- ❑ Formatting Definitions – Border, Style, Color
- ❑ Integration Definitions – Import Object, Import File
- ❑ Action Definitions – Key
- ❑ System Definitions

Interface Definitions

Definitions which are used in creating a user interface are referred to as an interface definition. The definitions in this category are Menu, Report, Form, Part, Line, Fields, Button and Table.

Menu: A Menu displays a list of options. The Tally.ERP 9 application determines the action to be performed on the basis of the Menu Item selected by the user. The 'Gateway of Tally' is an example of a 'Menu'. A Menu can activate another Menu or Report.

Report: This is the fundamental definition of TDL. Every screen which appears in Tally.ERP 9 i.e. the input screen or output screen is created using the report definition. A Report consists of one or more Forms.

Form: A Form consists of one or more Parts. Part: Part consists of one or more Lines.

Line: A Line consists of one or more Fields.

Field: A field is place where the data is actually displayed or entered. The data can be a constant or variable data.

Button: The user can perform an action in three ways i.e. by selecting a menu item, by pressing a key and by clicking on a button. The Button definition allows the user to display a button on the Button bar and execute an action.

Table: The Table definition displays a list of values as Tables. Data from any collection can be displayed as a Table.

Data Definitions

Definitions which are used for storing the data are referred to as a Data Definitions. The definitions in this category are Object, Variable and Collection.

Formatting Definitions

Definitions which are used in formatting a user interface are referred as Formatting Definition. Definitions in this category are Border, Style and Color.

Integration Definitions

Definitions which make the import of data available in SDF (Standard Data Format) are referred to as Integration Definitions. Import Object and Import File are the two definitions classified in this category.

Action Definitions

The action definition allows the user to define a action when a key combination is pressed. It also associates an object on which the action is performed. The Key definition falls in this category.

System Definitions

System Definitions are viewed as being created by the administrator profile. Any items defined under System Definitions are available globally across the application. System Definitions can be defined any number of times in TDL. The items defined are appended to the existing list. System Definitions cannot be modified. E.g. of System Definitions are System: Variable, System: Formula, System: UDF and System: TDL Names

2.2.2 Attributes & Modifiers

An attribute is a property or characteristic for the Definition. Each definition has different attributes. There is a predefined set of attributes provided by the platform for each definition type. The attribute specifies the behavior of a definition. A Definition can have multiple attributes associated with it. Each attribute has a 'Name'(predefined) and an assigned value (provided by the programmer).

Example:

```
[Part: PartOne]
    Line: PartOne
```

In the above example, **Line** is the name of attribute, specific for the definition type. **PartOne** attribute value, it can be any constant or a formula.

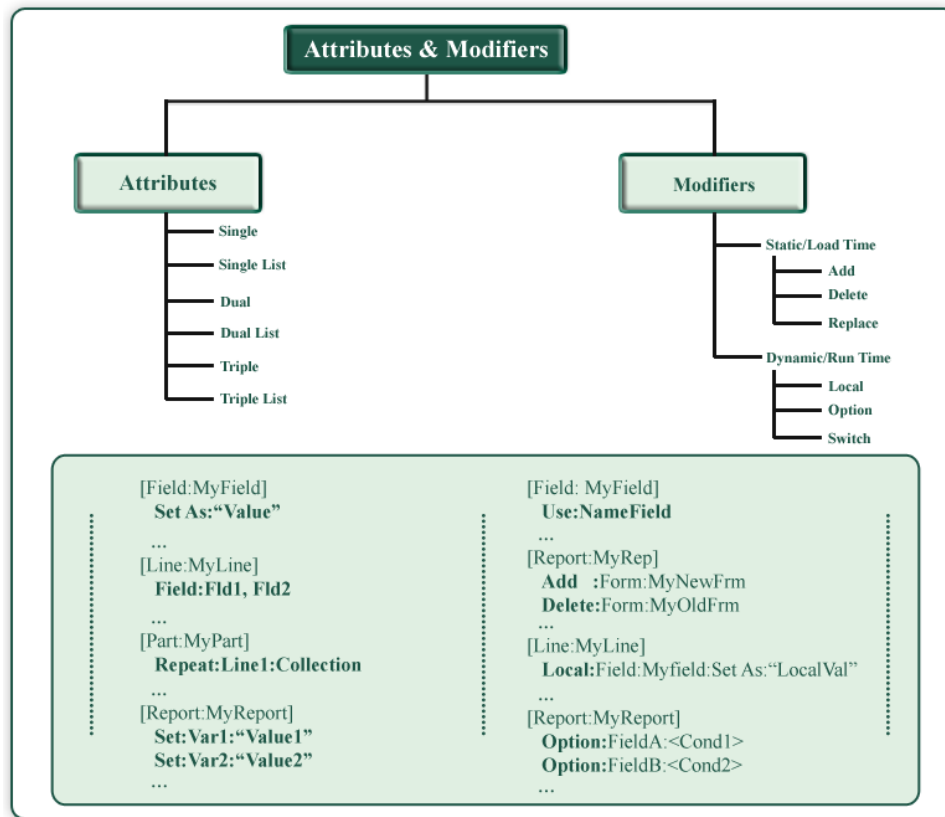


Figure 2.3 Attributes and Modifiers

Classification of Attributes

The classification of an attribute is done on the basis of the number of values it accepts and if they can be specified multiple times under the definition i.e. based on the number of sub attributes and the number of values. There are seven types of attributes, they are: Single and Single List, Dual and Dual List, Triple and Triple List and The Attribute type Menu Item.

Modifiers are used to perform a specific action on definition or attribute. They are classified as Definition Modifiers and Attribute Modifiers. Definition Modifiers are #, ! and *. Attribute Modifiers are Use, Add, Delete, Replace/Change, Option, Switch and Local. They are classified into two:

- Static/Load time modifiers: Use, Add, Delete, Replace/Change
- Dynamic/Real time modifiers: Option, Switch and Local

2.2.3 Data Types in TDL

The Data Types in TDL specify the type of data stored in the field. TDL being the business language supports business data types like amount, quantity, rate apart from the other basic types. The data types are classified as Simple Data Type and Compound Data Type.

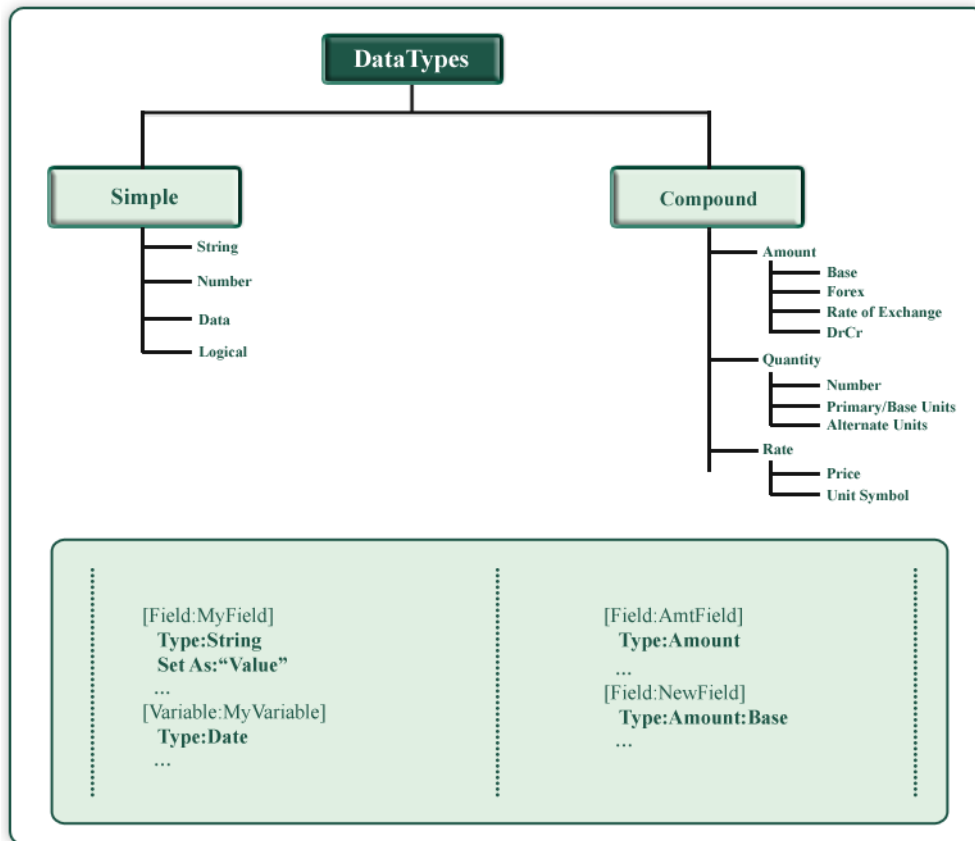


Figure 2.4 DataTypes

- Simple Data Type

This holds only one type of data. These data types cannot be further divided into sub-types. String, Number, Date and Logical data types fall in this category.

□ Compound Data Type

This is a combination of more than one data type. The data types that form a compound data type are referred to as sub-data type. The Compound Data types in TDL are: Amount, Quantity, Rate, Rate of exchange and Aggregate.

| Data Type | Sub-Types |
|---------------------|---|
| Simple Data Types | |
| Number | |
| String | |
| Date | |
| Logical | |
| Compound Data Types | |
| Amount | Base / Direct Base |
| | Forex |
| | Rate of Exchange |
| | DrCr |
| Quantity | Number |
| | Primary Units/ Base Units |
| | Secondary Unit/ Alternate Units/ Tail Units |
| Rate | Price |
| | Unit Symbol |
| Rate of Exchange | |

The type for the field definition is specified using the Type attribute.

Example:

```
[Field: Qty Secondary Field]
```

```
    Type : Quantity: Secondary Units
```

Qty Secondary Field is the name of the field, Type is the attribute name, **Quantity** is the datatype and **Secondary Units** is the subtype for the field

2.2.4 Operators in TDL

Operators are special symbols or keywords that perform specific operations on one, two or three operands and then return a result.

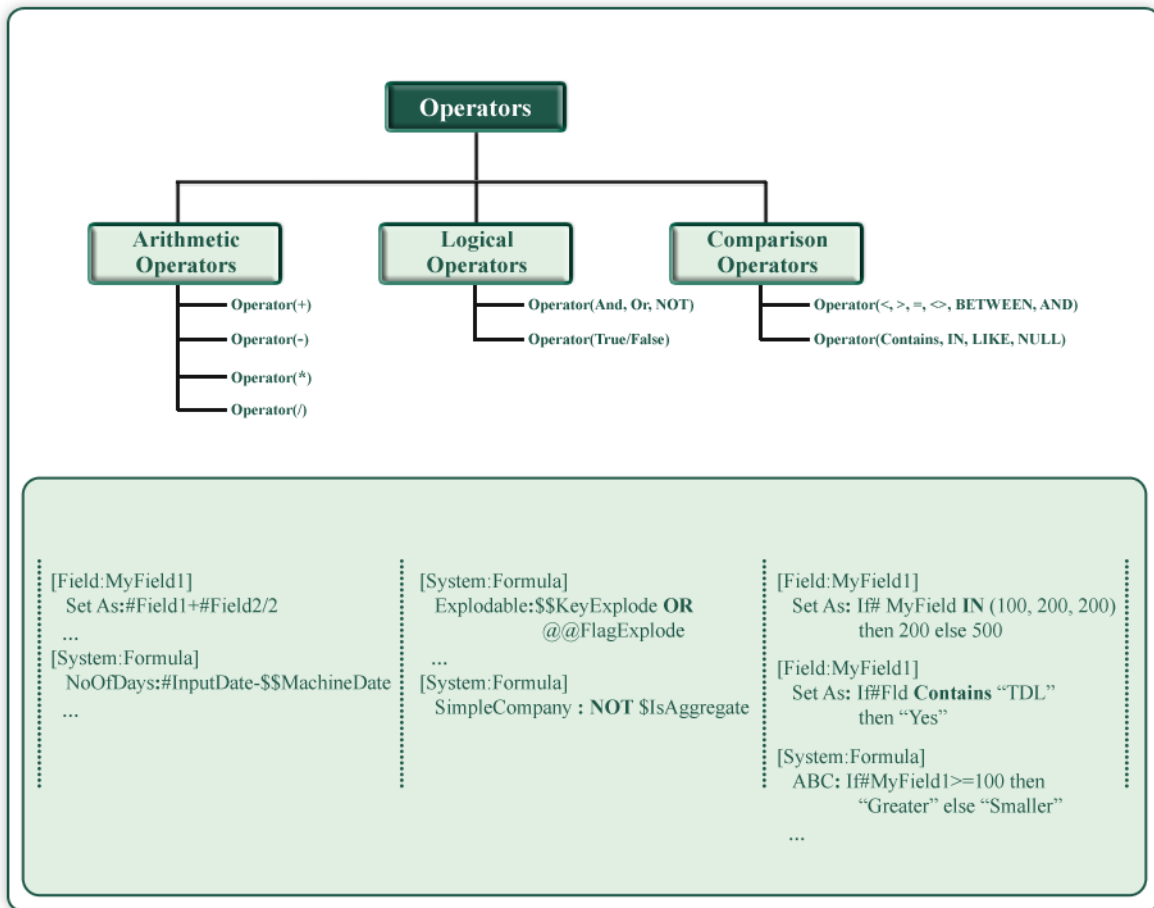


Figure 2.5 Operators in TDL

The three types of operators in TDL are as follows:

- ❑ **Arithmetic Operators** - The arithmetic operators supported by TDL are + (Addition), - (Subtraction), / (Division) and * (Multiplication).
- ❑ **Logical Operators** - The logical operators used are: OR, AND, NOT, TRUE/ON/YES and FALSE/OFF/NO
- ❑ **Comparison Operators** - A comparison operator compares its operands and returns a logical value based on whether the comparison is true.
- ❑ **String Operators** - A string operator allows comparing two strings.



The operator = is a comparison operator, not assignment operator. There is no assignment operator in TDL.

While evaluating the expression some keywords are ignored. The keywords which are not considered are Than, With, By, To, Is, Does, Of.

2.2.5 Special Symbols

The Symbol Prefix in Tally Definition Language (TDL) has different usage and behavior when used with different Definitions and attributes of definitions.

The different special symbols used in TDL are \$, \$\$, @, @@, #, ##, ;, ;;, :::, /* */, +, !, * and _ . Each of these symbols are used for a specific purpose. The usage of each of these symbols will be discussed in detail in the subsequent chapters.

2.2.6 Actions in TDL

Actions are activators of specific functions with a definite result. TDL is an action driven language. TDL does not have a procedural flow of control. In TDL the user will do some operation in Tally application, and then some action got executed. In TDL actions are used to implement or activate the definitions. For example, Display is an action which is used as below:

```
[#Menu: Gateway of Tally]
  Add      : Item      : Action Reports: Display  : Action Report
```

In Tally application, the Ctrl+A Key pressed from a voucher accepts the Entry Screen. Examples of Actions are: Display, Menu, Print, Create, Alter etc.

Example:

```
Action: Create: My Sample Report
```

Action is the Keyword

Create is the name of the action to be performed. It can be any of the pre-defined actions.

My Sample Report is the name of the definition/variable on which the specified action is to be performed.

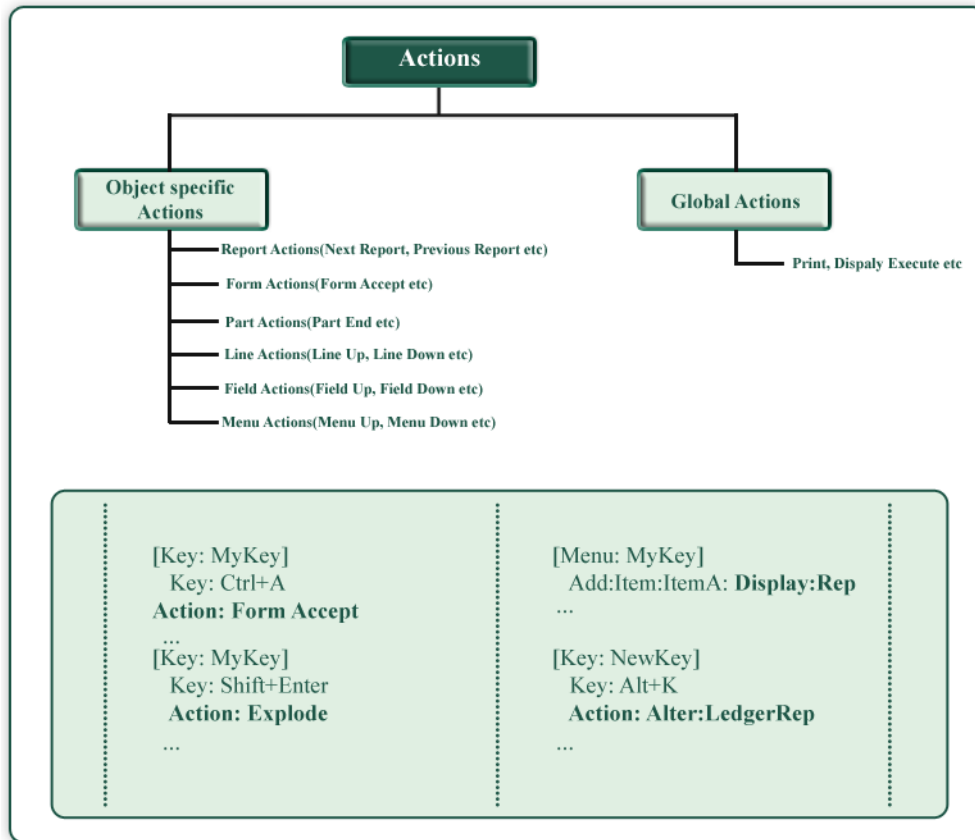


Figure 2.6 Actions in TDL

2.2.7 Functions

The functions in TDL are defined and provided by the platform. These functions are meant to be used by the TDL programmer and are specifically designed to cater to the business requirement of the Tally.ERP 9 application. TDL has a library of functions which allows performing string, date, number and amount related operations apart from the business specific tasks. Some of the basic functions provided by TDL are `$$StringLength`, `$$Date`, `$$RoundUp`, `$$AsAmount` etc.

Example:

```
$$SysName:EndOfList
```

`$$Sysname` is the Function name and `EndOfList` here acts as an argument.

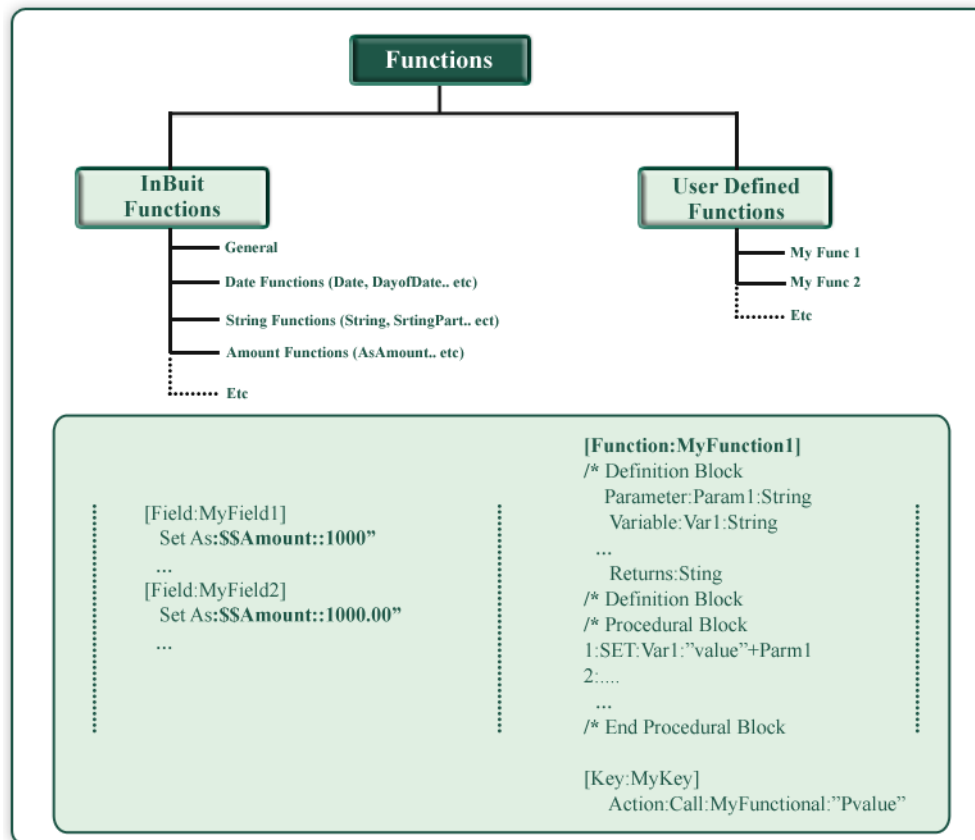


Figure 2.7 Functions in TDL

2.3 Variables in TDL

A Variable is a storage location or entity. It is a value that can change, depending on the conditions or on the information passed to the program. In TDL, a Variable helps to control the behavior of Reports and their contents. Variables assume different values during execution and these values affect the behavior of the Reports. A Variable definition is similar to any other definition.

2.3.1 Types of Variable

The variables are classified into different types based on the type of values that they can store. In TDL, the variables are classified into four:

- ❑ **Simple Variable** - A variable which can store single value
- ❑ **Repeat Variable** - A variable which can store multiple values of a single data type within the Report Scope
- ❑ **List Variables** - A variable which can store multiple values of a single datatype identified by a 'Key'. Key provides the flexibility of accessing a value directly from the List
- ❑ **Compound Variables** -Variable which can store multiple values of different data types.

Consider the following example to understand the concept of variable in TDL.

```
[Report : Variable Demo]
  Form      : Variable Demo
  Variable  : SampleVar
```

In the above definition, declaring the variable as local by attaching at the Report level using the attribute **Variable**. **SampleVar** is the variable name.

```
[Field: Initial Value]
  Use       : Name Field
  Width    : 35
  Modifies  : Sample Var
```

```
[Field: Value from Variable]
  Use       : Name Field
  Width    : 35
  Skip     : Yes
  Set As   : ##SampleVar
```

Above two field definitions, **initial Value** and **Value from Variable**. In the first definition modifying the Variable value to the user entered value using the attribute **Modifies**. Since variable 'Sample Var' is a local variable to this report, modifying in this field will not retain the value once the report is quit.

The second field which sets the value of variable as Sample Var.

```
[Variable: SampleVar]
  Type      : String
  Default   : "Default Value"
```

In the above code snippet gives the actual definition of the Variable. **SampleVar** is the Variable name. **Type** and **Default** are the attributes of variable. Following section gives a brief explanation about various attributes of variable definition.

2.3.2 Attributes of Variable Definition

The attributes that defines the behavior of the variable are Type, Default, Volatile and Persistent.

Type

The data type of the variable is specified using the Type attribute of variable definition.

All the data types supported by TDL can be used to specify the variable data type. The Types of values that a variable can handle are String, Logical, Date and Number. The Amount, Quantity and Rate data type can be used in the context of company object only.

In the absence of attribute Type, String is considered as the default data type.

Example:

```
[Variable: Test Var]
  Type      : Logical
```

Test Var is the variable name, **Type** is the variable attribute and **Logical** is the data type defined for the variable Test Var.

Default

This attribute provides the initial value for the variable. The initial value can be overridden while defining the system scope.

Example:

```
[Variable: Test Var]
  Type      : Logical
  Default   : Yes
```

The default value of the variable '**Test Var**' is set to **Yes**.

Persistent

This attribute allows storing the value of variable permanently across sessions. The value of the variable modified by the last session is retained even after exiting the application. Only the variable with the system scope can be persistent.

Example:

```
[System: Variable]
  Test Var   : 'No'

[Variable: Test Var]
  Type      : Number
  Persistent : Yes
```

The latest value of variable Test Var will be retained by the application.

Volatile

The attribute Volatile, when set to Yes, allow the user to store multiple values of the variable in the stack. The default value of this variable is Yes.

It stores multiple values within the report scope. When a report is called from current report, the variable will store retain the both the values modified through the current report and the previous report. When user exits the current report the value is restored to the previous one.

Example:

```
[Variable: Test Var]
  Type          : Number
  Volatile      : Yes
```

The variable Test Var is specified in the report R1 which sets its value as 10. The report R2 is called from R1, and it sets the value to 20. Further a report R3 is called from R2 which modifies the value as 30. At this point in time, the Test var variable holds three values in stack 10, 20, 30.

When user quits the R3 report, the variable restores the value to 20 and so on.

The actions in TDL can be delivered in three ways:

- By activating a Menu Item
- By pressing a Key
- By activating a Button.

Button /Key is a definition 'Button' refers to the button that appears on the button bar, on clicking the relevant button, the associated action gets triggered. The action to be associated with this button needs to be defined in TDL. The key refers to the key combination that is associated along with the button. The action can be triggered by pressing that key combination also.

Lesson 3: Reports in TDL

Lesson Objectives

On completion of this lesson, you will be able to

- Understand how to display the Tally.ERP 9 reports using TDL
- Understand the dimensions and formatting of Reports
- Understand the concept Object
- Understand how to create and edit reports using TDL

Introduction

In Tally.ERP 9 the financial statements are generated as Reports based on the vouchers entered till date. The Balance Sheet, Profit & Loss A/c, Trial Balance etc are the some of the Reports which Tally.ERP 9 has by default. Normally a business requires Reports in any of the following formats:

- **Tabular Report:** A Report with fixed number columns which can be configured
- **Hierarchical Report:** A Report designed in successive levels or layers
- **Columnar Report:** A Report with multiple columns

Tally.ERP 9 caters to generating all the types of Reports mentioned above.

3.1 Display Reports

A menu item defined with Display action is used to display a report. Consider the menu item **Quarterly Report** Item defined in the menu **Gateway of Tally**. To display a report **Quarterly Report** using the menu item **Quarterly Report** item, use the following code.

```
[#Menu: Gateway of Tally]
```

```
Item: Quarterly Report Item: Display: Quarterly Report
```

3.1.1 Understanding Dimensions & Formatting

Dimensions are specifications. Dimensions in TDL are effective either in the display mode or in the print mode. The data in TDL does not have an absolute position of the dimensions specified but a relative. There are four definitions in TDL that attract dimensions. They are: Form, Part, Line and Field.

In TDL the following Unit of Measurements can be used:

Millimeters/ mms, Centimeters/ cms, Inch (es), Number of Characters/ Number of Lines, % Screen/ Page, Number - Points (where 1 Point = 1/72 Inch)

Dimensional Attributes

Dimensional Attributes can be classified into two i.e., Specific and General Attributes.

| Definitions | Specific Dimensions | General Dimensions |
|-------------|---|---|
| Form | Height, Width, Space Top, Space Bottom, Space Left, Space Right | Horizontal Align, Vertical Align, Full Height, Full Width |
| Part | Height, Width, Space Top, Space Bottom, Space Left, Space Right | Horizontal Align |
| Line | Height, Space Top, Space Bottom, Indent | Full Height |
| Field | Width, Space Left, Space Right, Indent | Full Width, Widespaced |

Definitions and Attributes for Formatting

Definition - Border

The Definition Border determines the type of lines required in a border which can be used by a Part, Line or a Field which means that this definition can define customized borders for the user. But it is ideal to use the predefined borders which are part of the default TDL instead of user defined, since almost all possible border combinations are already defined in the Default TDL. Top, Bottom, Left, Right, Color and PrintFG are the attributes of Border definition.

Consider the following example:

```
[Field: Sample Border and Style]
Set as      : "Welcome 2 Tally"
Width      : 60 mms
Border     : Sample Border
```

```
[Border: Sample Border]
  Top      : Thin,Double
  Bottom   : Thick
  Left     : Flush
  Right    : FullLength
  Color    : Red
```

Definition - Style

The Definition Style can be used in the Field Definition only. This definition determines the appearance of the text being displayed/printed by using a corresponding font scheme, Bold, Italic, Point Size, Font Name, etc.

Style Attribute in Field Definition is used to format the appearance of the text appearing within the Field both in display and print mode provided the Print Style Attribute is not used within the current Field. Print Style attribute in Field is used if the Style required while displaying is different from the Style required while printing. Consider the following example:

```
[Field: Sample Border and Style]
  Set as      : "Welcome 2 Tally"
  Width      : 60 mms
  Style       : Sample Style
  Border      : Sample Border

[Style: Sample Style]
  Font        : "Algerian"
  Height      : 16
  Bold        : Yes
```

Definition - Color

The Definition Color is useful to determine the Foreground and Background color for a Form, Part, Field or Border both in Display as well as Print Mode,

Color specification can be done by specifying the RGB Values (the combination of Red, Green and Blue - each value should range from 0 to 255)

For example:

```
[Color: Light Grey]
  RGB      : 218, 218, 218
```


Consider the following example for the better understanding of dimensions

```
/*
```

```
This program illustrates different attributes used for formatting and border and style definition and its attributes
```

```
*/
```

```
[Report: Dimension Sample]
```

```
Form           : Dimension Sample
```

```
[Form: Dimension Sample]
```

```
Parts          : Dimension Sample
```

```
[Part: Dimension Sample]
```

```
Left Part      : LRPP Part1
```

```
Right Part     : LRPP Part2
```

```
[Part: LRPP Part1]
```

```
Lines         : LRPP Part1 Line
```

```
[Line: LRPP Part1 Line]
```

```
Fields: LRPP Part1 Field1, LRPP Part1 Field2
```

```
[Field: LRPP Part1 Field1]
```

```
Set as        : "Welcome to the World of TDL - Left"
```

```
Style         : Sample Style
```

```
Border        : Sample Border
```

```
[Field: LRPP Part1 Field2]
```

```
Set as        : "Hello TDL - Left"
```

```
Style         : Sample Style
```

```
Border        : Sample Border
```

```
[Part: LRPP Part2]
```

```
Lines        : LRPP Part1 Line
```

```
Local        : Field: LRPP Part1 Field1: Set As: "Welcome to the World of TDL -  
Right"
```

```
Local        : Field: LRPP Part1 Field2: Set As: "Hello TDL - Right"
```

;; Border Definition

```
[Border: Sample Border]

Top      : Thin,Double
Bottom   : Thick
Left     : Flush
Right    : FullLength
Color    : Red
```

;; Style Definition

```
[Style: Sample Style]

Font     : "Algerian"
Height   : 16
Bold     : Yes
```

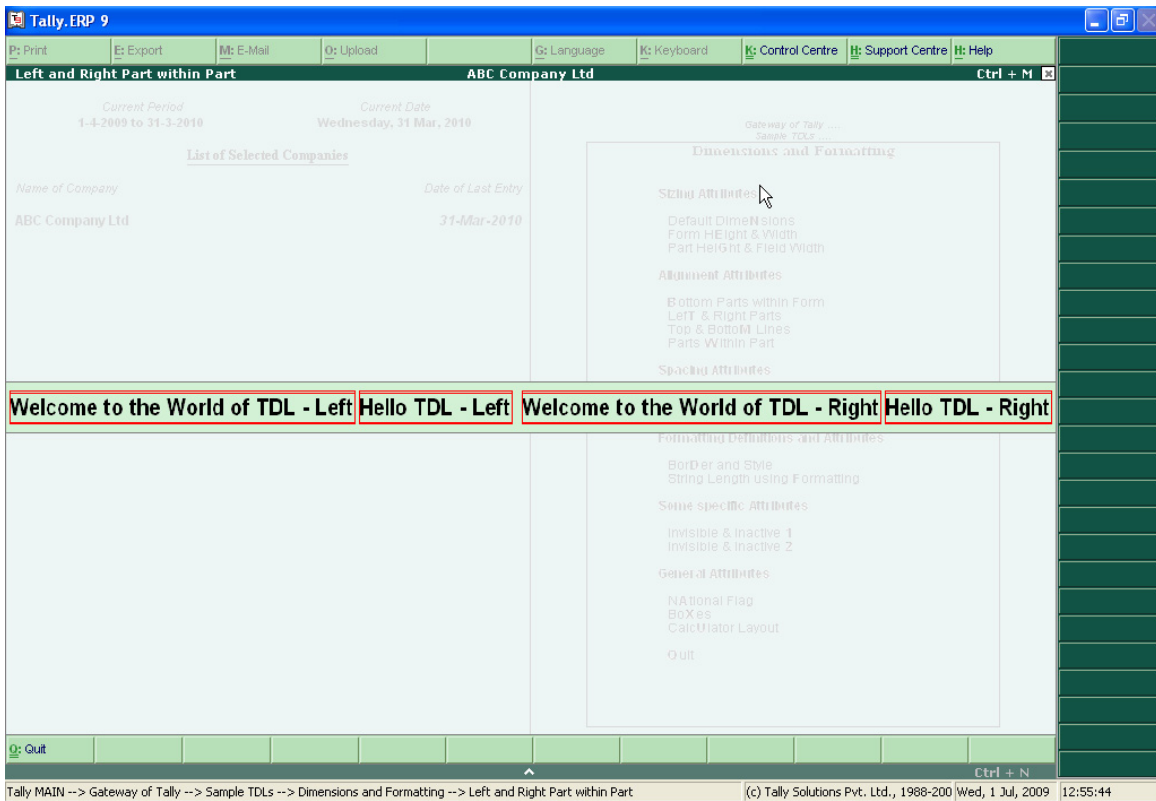


Figure 3.1 Dimensions

3.1.2 Tally Data Structure- Objects, Methods and Collections

Any information that is stored in a computer is referred to as Data. Database is a collection of information organized in such a way that a computer program can quickly select desired data. A database can be considered as an electronic filing system. To access information from a database a Database Management System (DBMS) is used. DBMS allows entering, organizing, and selecting data in a database.

The organization of data in a database is referred to as the 'Database Structure'. The widely used database structures are hierarchical, relational, network and object-oriented. In the hierarchical structure the data is arranged in a tree like structure. This structure uses the parent - child relationships to store repeating information. A parent can have multiple children but a child can have only one parent. The child in turn can have multiple children. Information related to one entity is referred to as an object. A database is a group of interrelated objects.

An object is a self-contained entity that consists of both data and procedures to manipulate the data. It is defined as an independent entity based on its properties and behavior/functionality. Objects are stored in a data base. A relationship can be created between the objects. As discussed, the hierarchical structure has a parent-child relationship. For example, child objects can inherit characteristics from parent objects. Likewise, a child object can not exist with out a parent object. After discussing the object concept in general, let us examine the Tally object structure in the following section.

Tally Object Structure

The Tally data base is hierarchical in nature in which the objects are stored in a tree like structure. Each node in the tree can be a tree in itself. An object in Tally is composed of methods and collection. Method is used to retrieve data from the database. A collection is a group of objects. Each object in the collection can further have methods and collection. The structure is as shown in Figure.

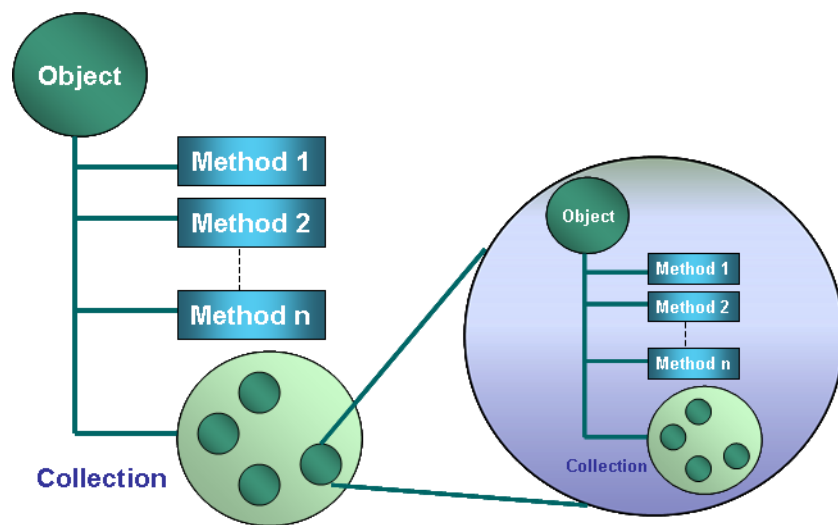


Figure 3.2 Object Structure

Everything in TDL is an Object. As mentioned in the earlier chapters, Report, Menu, Company, Ledger all are objects in TDL. The properties of objects in TDL are called as attributes. For example, the attributes Object, Title, Form are all properties that define the Report object.

For example, the Object Ledger contains the Methods Name, Parent etc. and the collections Address and Billwise Details.

As shown in the above figure Objects available at Level 1 are referred as Primary objects and objects which are at Level 2-n are referred as secondary objects.

The types of Objects can be broadly classified into two:

- **Interface Objects** - It defines the user interface while Data objects store the value in the Tally Primary or Secondary database. Any data manipulation operation on the data object is performed through Interface objects only
- **Data Objects** - Data is actually stored in the Data Objects. These objects are classified into two types viz., Internal objects and User defined objects / TDL objects.
- **Internal objects** - Internal objects are provided by the platform. They are stored in tally database. Multiple instances of internal objects can exist. In Tally, internal objects are of several types. Examples of internal objects are Company, Group, Ledger, Stock, Stock Item, Voucher Type, Cost Centre, Cost Category Budget, Bill and Unit of Measure.
- **User Defined Objects /TDL Objects** - All Objects which are defined by the user in TDL are referred as User Defined Objects or TDL objects. User defined objects are further classified as Static Objects or Dynamic Objects. Static TDL objects cannot be stored in Tally database. The data for the static object is hard coded in the program and can be used for display purpose only.

The dynamic TDL objects can be created from the data available in any of the following external data sources:

- XML Files from remote HTTP server
- DLL files
- From any type of database through ODBC

For example, an Employee Object can be created, which gives Name, ID and Designation. These values can be provided as Methods. All such Objects can be put together inside a Collection. The following sample code illustrates this.

```
[Object: Employee1]
  Name: "Ashok"
  EmpID: 101
  EmpDesignation: Trainee-Developer
```

```
[Object: Employee2]
  Name: "Anad"
  EmpID: 99
  EmpDesignation: HR-Executive
```

```
[Collection: Employee Masters]
```

```
Object: Employee1
```

```
Object: Employee2
```

Methods

Each piece of information stored in the data object can be retrieved using a method. A method either performs some operation on the object or retrieves a value from it. To retrieve the value from the database, the storage name is prefixed with the \$ symbol. TDL provides pre-defined Methods and allows the user to create methods as well. Methods are classified as Internal or External methods.

- **Internal methods** are predefined methods. For example, under the object ledger Methods, Name, Opening Balance etc. are internally defined.
- **In External Methods** A user can change the behavior or perform an action on the internal object by defining new Methods. Methods defined by the user are referred to as External methods or User defined methods.

Example:

```
[#Object: Bill]
```

```
BalanceDiffAmt: $OpeningBalance - $ClosingBalance
```

In the above code, **OpeningBalance** and **ClosingBalance** are default Methods. **BalanceDiffAmt** is a User Defined Method which gives the difference of **Opening Balance** and **Closing Balance** for the Bill. The user can access this method using **\$BalanceDiffAmt**.

Collection

A collection can be a collection of objects or a collection of collections. The collection of collections is referred as Union of collection. In TDL collections are of two types: Simple collection and Compound collections.

- **Simple Collection:** Simple collections only have a single method which is repeatable. Simple Collections cannot have sub-collections. The Name and Address are examples of Simple Collections.
- **Compound Collections:** The collections which have sub-collections and multiple methods are called Compound Collection. Any Internal or External Collections of Primary or Secondary or user defined objects is an example of a Compound Collection.

Collection, the data processing artifact of TDL provides extensive capabilities to gather data not only from Tally database but also from external sources using ODBC, DLLs and HTTP.

Creating Employee Collection

Collections can also be a cluster of External Objects. Consider the following example of **Employee Collection** of External Objects.

```
[#Menu: Gateway of Tally]
    Add: Item : Employee Dept. : Display :Employee List

[System: Formula]
    EmpHRA      : $EmpBasic * 0.10
    AmtWidth    : 10

[Report: Employee List]
    Form       : EmpForm

[Form: Emp Form]
    Part      : EmpPart

[Part: Emp Part]
    Line           : Emp Titles, Emp Line
    Repeat         : Emp Line : MyEmps
    Scroll         : Vertical
    Common Border  : Yes

[Line: Emp Titles]
    Use      : Emp Line
    Local : Field : FEmp Name : Set as: "Employee Name"
    Local : Field : FEmp Basic : Type : String
    Local : Field : FEmp HRA  : Type : String
    Local : Field : FEmp Salary : Type : String
    Local : Field : FEmp Basic : Set as: "Basic"
    Local : Field : FEmp HRA  : Set as : "H.R.A."
    Local : Field : FEmp Salary : Set as : "Salary"
    Local : Field : Default   : Align : Centre
    Local : Field : Default   : Style  : Normal Bold
    Border: Thin Top Bottom
```

[Line: Emp Line]

Field : FEmp Name
Right Fields: FEmp Basic, FEmp HRA, FEmp Salary

[Field: FEmp Name]

Set as : \$EmpName
Width : @@NameWidth
Space Left : 3

[Field: FEmp Basic]

Type : Amount
Set as : \$\$AsAmount:\$EmpBasic
Width : @@AmtWidth
Align : Right
Border : Thin Left Right

[Field: FEmp HRA]

Use : FEmp Basic
Set as : \$\$AsAmount:@@EmpHRA
Border : Thin Right

[Field: FEmp Salary]

Use : FEmp Basic
Set as : #FEmpBasic + #FEmpHRA
Border : Thin Right

[Collection: MyEmps]

Objects : Emp01, Emp02, Emp03, Emp04, Emp05

[Object: Emp01]

EmpName : "Ajay"
EmpBasic : 1000
EmpDept : "Training"

[Object: Emp02]

EmpName : "Anil"

EmpBasic : 2000

EmpDept : "Training"

[Object: Emp03]

EmpName : "Ashish"

EmpBasic : 3000

EmpDept : "Large Enterprise"

[Object: Emp04]

EmpName : "Sanjay"

EmpBasic : 4000

EmpDept : "Large Enterprise"

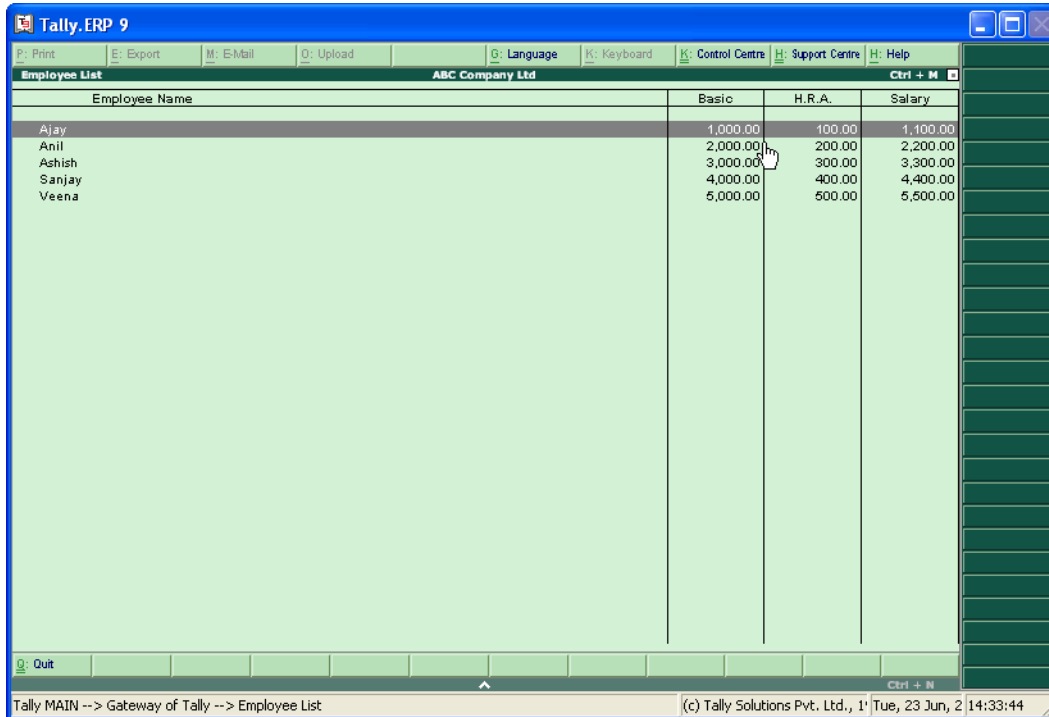
[Object: Emp05]

EmpName : "Veena"

EmpBasic : 5000

EmpDept : "Large Enterprise"

The output for the above program is as shown below.



| Employee Name | Basic | H.R.A. | Salary |
|---------------|----------|--------|----------|
| Ajay | 1,000.00 | 100.00 | 1,100.00 |
| Anil | 2,000.00 | 200.00 | 2,200.00 |
| Ashish | 3,000.00 | 300.00 | 3,300.00 |
| Sanjay | 4,000.00 | 400.00 | 4,400.00 |
| Veena | 5,000.00 | 500.00 | 5,500.00 |

Figure 3.3 External Object Output

3.2 Edit Reports

Create and Alter action acts only upon the Report Definition. These actions activate the Report in Create or Alter Mode. In other words, the Report is started in the Edit Mode. In case of Create Action, the user enters the Report in order to add values whereas in case of Alter, the user enters the Report to modify the already created values.

To store the values as a part of Tally Database, the Report must be associated to a Data Object. For example, Group, Ledger, Voucher, etc. are some of the Data Objects available in Tally.

For instance, in order to design an interface to create a Ledger:

- ❑ The Object Ledger must be associated to the Report using Report Attribute Object
- ❑ Values entered by the user in the Fields within the Report must be stored in relevant Methods using Field Attribute, Storage

Example:

The following code demonstrates the usage of Action Create and Attribute Storage at Field definition to store the values entered within the relevant Object associated at Report Level

```
[#Menu: Gateway of Tally]
  Add      : Key Item: Ledger Creation: L : Create:Create Ledger

;; Object Association done at Report Level
[Report: Create Ledger]
  Form     : Create Ledger
  Object   : Ledger

[Form: Create Ledger]
  Parts    : Create Ledger

[Part: Create Ledger]
  Lines    : Store LedgerName, Store LedgerGroup

[Line: Store LedgerName]
  Fields: Short Prompt, Name Field
  Local : Field: Short Prompt: Info: "Name :"
  Local : Field: Name Field: Storage: Name

;; Storing the value entered by user in an Internal Method Name available within the Object associated at the Report Level
[Line: Store LedgerGroup]
  Fields      : Short Prompt, Name Field
  Local       : Field      : Short Prompt: Info      : "Under :"
  Local       : Field      : Name Field: Storage    :Parent
  Local       : Field      : Name Field: Table      :Group
```

Similarly, Parent Method is stored with the user entered value which is considered as the Group of the Ledger created. Also Group is a default Table/ Collection to display all the default as well as user defined Groups. Field Attribute Table helps to restrict the user input to a predefined list.

In the above example,

- ❑ Default Menu, **Gateway of Tally** have been altered to add a new Item **Ledger Creation** which allows the user to create Ledger
- ❑ Report **Create Ledger** associates the Object Ledger to it which indicates that the Report is meant for creating an instance of the Object Ledger.
- ❑ The Name and Group of the Ledger are stored in the Internal Methods **Name** and **Parent** which stores.

Example:

The following code demonstrates the usage of Alter Action at Button

```
[Button: My Reco Button]
```

Button meant to do Bank Reconciliation

```
Key      : Alt + F5
```

```
Action: Alter: Bank Recon
```

Alter Action to trigger Bank Recon Report in Alter Mode

```
Title : "Reconcile"
```

```
[Form: My Bank Vouchers]
```

```
Button: My Reco Button
```

Associating the Button to the Report

In the above example,

- Button **My Reco Button** is defined with **alter** action to alter Report **Bank Recon** on pressing **Alt + F5** Key. *This button is used for entering dates in Bank Reconciliation Report.*
- Button **My Reco Button** is associated to the Form **My Bank Vouchers**

3.3 Designing Reports using Existing Data

3.3.1 Simple Trial Balance Report

Design a report which prints ledger name and closing balance as follows:

ABC & Company 12,000.00

Cash A/C 6,000.00

State Bank of India 12,000.00

Use the following code.

```
[#Menu: Gateway of Tally]
```

```
Add      : Item: MyTrialBalance : Display : My Trial Balance
```

```
[Report: My Trial Balance]
```

```
Form      : My Trial Balance
```

```
[Form: My Trial Balance]
```

```
Top Part  : MyTB Detail
```

```
[Part: MyTB Detail]
```

```
Top Line  : MyTB Detail
```

Repeat : MyTB Detail: MyTB Collection
Scroll : Vertical

[Line: MyTB Detail]

Left Fields : MyTB Name

Right Fields : MyTB Amt

[Field: MyTB Name]

Width : 30

Set as : \$Name

[Field: MyTB Amt]

Type : Amount

Width : 15

Set as : \$ClosingBalance

Align : Right

[Collection: MyTB Collection]

Type : Ledger

- **The Set as: \$Name** and Set as: **\$ClosingBalance** are Internal Methods which are associated with the Fields, **MyTB Name** and **MyTB Amt**, under the Line **MyTB Detail** with Collection **MyTBCollection**.
- The **Repeat** attribute at the Part level repeats the Top Line **MyTB Detail** to list all the Objects under the Collection **MyTB Collection**.
- Attribute **Scroll: Vertical** is used under the **Part** because it increases the visibility of the Lines and scrolls the screen without making the Lines shrink to fit the screen. This is useful if the Trial Balance of a Company has a considerable number of ledgers.

The output of the program as shown:

| Simple Trial Balance | | ABC Company Ltd | Ctrl + M |
|-------------------------------------|--|-----------------|----------|
| ABC India Pvt. Ltd. | | 32,650.00 | |
| Accum. Dep. on Airconditioner | | 16,300.00 | |
| Accum. Dep. on Building | | 1,40,000.00 | |
| Accum. Dep. on Computer & Per. | | 36,184.00 | |
| Accum. Dep. on Furn. & Fixt. | | 12,242.00 | |
| Accum. Dep. on Motor Car | | 1,81,146.20 | |
| Accum. Dep. on P & M - I | | 1,50,000.00 | |
| Accum. Dep. on P & M - II | | 78,750.00 | |
| Advance Tax | | 2,75,000.00 | |
| Advertisement Charges | | | |
| Advisory Consultants | | 15,000.00 | |
| Airconditioner | | 21,500.00 | |
| All India Computer Institute | | 78,320.00 | |
| Amar Computer Peripherals | | 12,69,680.00 | |
| Anvind Kumar | | 508.00 | |
| Ashok Financiers | | 1,50,000.00 | |
| Assembling Charges | | | |
| Avanathi Constructions | | | |
| AVT Computers | | 58,388.00 | |
| Balasubramanian's Share Capital A/c | | 7,13,000.00 | |
| Bank Charges | | | |
| Bank of India | | | |
| Basic Pay | | | |
| Batliwala & Co. | | | |
| Best Systems Pvt Ltd | | | |
| B. Ganesh | | | |
| Bharat Petroleum | | 1,520.00 | |
| Bonus | | | |
| Bonus Paid | | | |
| B Ramesh - Loan | | | |
| Building | | 6,00,000.00 | |
| | | 224 more ... | |

Figure 3.4 Trial Balance - Output

3.3.2 Tabular Reports

A Tabular Report has the simplest format of all the Report formats. A typical Tabular Report will have following components:

- Report Title: It contains the Name of the Report, the Title for each column, the Day/ Period for which a Report is generated, etc.
- Report Details: It contains the actual information
- Report Total: It contains the Total of the respective columns

In a typical Tabular Report, the number of columns is fixed and is interactive i.e. an end user can change the appearance of the Report. The Day Book, Stock Summary, Trial Balance, Group Summary are the some of the default Tabular Reports in Tally.ERP 9.

Designing a Tabular Report

A typical Tabular Report will have a Title Line, Details Line and an optional Total Line. The Details Line will be repeated over the objects of a Collection. A Tabular Report can be made Interactive by adding the following features.

- Adding Buttons to change the period, to change the contents of the Report, etc.
- Adding explosions to the lines

Example: Simple Tabular Report

Let us consider writing a simple Trial Balance.

```
[Part: My TB Part]
  Lines      : My TB Title, My TB Details
  Repeat     : My TB Details: My TB Groups
  Scroll     : Vertical
```

The following code snippet displays the exploded part:

```
[Line: My TB Details]
  Fields      : My TB Name Field, My TB ParName Field
  Right Fields: My TB Dr Amt Field, My TB Cr Amt Field
  Explode     : My TB Group Explosion : $$IsGroup and $$KeyExplode

  [Field: My TB Name Field]
    Use       : Name Field
    Set as    : $Name
    Variable: MyGroupName1
```

The code for the exploded part is as shown below:

```
[Part: My TB Group Explosion]
  Lines      : My TB Details Explosion
  Repeat     : My TB Details Explosion : My TB GroupsLedgers
  Scroll     : Vertical
```

```
[Line: My TB Details Explosion]
  Fields      : My TB Name Field, My TB ParName Field
  Right Fields : My TB Dr Amt Field, My TB Cr Amt Field
  Explode     : My TB Group Explosion : $$IsGroup and $$KeyExplode
  Indent      : 2 * $$ExplodeLevel
  Local      : Field : Default : Delete: Border
```

In the code snippet, the Collection **My TB GroupLedgers** is a union of collections of the Type Group and Ledgers respectively.

```
[Collection: My TB GroupsLedgers]
  Collection : My TB Groups, My TB Ledgers
```

The variable **MygroupName1** is used in the attribute Child Of under the collections My TB Groups and My TB Ledgers.

```
[Collection: My TB Groups]
  Type      : Group
  Child Of  : #MyGroupName1
[Collection: My TB Ledgers]
  Type      : Ledger
  Child Of  : #MyGroupName1
```

When the user presses the Shift + Enter keys, then the exploded part shows the Sub-groups under the group in the current line.

Attributes and Functions used in the above program

The attribute Explode

The attribute 'Explode' refers to an attribute of the line, which is used to take the current data from the Line Object. A Part is displayed after the process of explosion is complete.

Syntax

```
Explode : <Part Name> : <Logical Condition>
```

<Part Name> is the name of the Part which displays the additional information about the Line object.

<Condition> If the Condition is True, then it will result in an explosion.

The Function of \$\$KeyExplode

\$\$KeyExplode function gives the current status of the keys Shift and Enter. This is used as a condition to check if the user has pressed the Shift+Enter Keys.

3.3.3 Hierarchical Report (Drill down Report)

A Tally application provides a simple way of navigating from one report to another which is commonly referred to as a drill down. A Drill Down facility moves from one report to the other to give a detailed view based on the selection in the current report. A user can return to the first Report from the detailed view. A typical drill down in Tally.ERP 9 starts from the Report and reaches the Voucher Alteration screen.

Designing Hierarchical Reports

A Hierarchical Report can be designed by incorporating the following changes to a Tabular Report.

- ❑ Variable attribute of Report definition
- ❑ Child Of attribute of Collection definition
- ❑ Display and Variable attributes of Field definitions
- ❑ Variable Definition

Example:

The following code snippet demonstrates the Drill down action, which is based on the Group Name displayed in the field. The Drill down action is achieved by specifying the two attributes Variable and Display at the field level.

```
[Field: MyTB Name]
  Width      : 120 mms
  Set as     : $Name
  Variable   : GroupVar
  Display    : My Trial Balance : $$IsGroup
```

A Variable is defined as a Volatile and is associated at Report. The attribute Variable of Report definition is used to associate the Variable with the report.

```
[Variable: Group Var]
  Type       : String
  Default    : ""
  Volatile   : Yes
```

```
[Report: My Trial Balance]
  Form       : My Trial Balance
  Variable   : GroupVar
```

The same Variable is used in the Childof attribute of the Collection definition. When a line is repeated over this collection in the report when the user presses the Enter key the Report being displayed will have the objects whose Parent Name is stored in the variable.

```
[Collection: My Collection]
  Type       : Group
  Childof    : ## GroupVar
```


The following screen is displayed when the user selects the option from the Menu:

| My TB Report | | ABC Company Ltd | | Ctrl + M |
|---------------------|---------|-----------------------|-----------------------|----------|
| Name | Parent | Debit | Credit | |
| Capital Account | Primary | | 55,00,000.00 | |
| Current Assets | Primary | 3,16,47,171.92 | | |
| Current Liabilities | Primary | | 51,11,656.90 | |
| Fixed Assets | Primary | 34,37,489.68 | | |
| Indirect Expenses | Primary | | 14,500.00 | |
| Investments | Primary | 5,00,000.00 | | |
| Loans (Liability) | Primary | | 27,27,116.03 | |
| Sales Accounts | Primary | | 6,05,000.00 | |
| TOTAL | | 3,55,84,661.60 | 1,39,58,272.93 | |

Figure 3.5 Simple Trail Balance Report

When the key **Enter** is pressed by the user, the next screen displays the Sub Groups of the current Group

| Trial Balance | | ABC Company Ltd | | Ctrl + M |
|--------------------------|-----------------------|-----------------------|--------|----------|
| Name | Parent | Debit | Credit | |
| Bank Accounts | Current Assets | 38,35,633.77 | | |
| Cash-in-hand | Current Assets | 1,83,62,572.24 | | |
| Loans & Advances (Asset) | Current Assets | 16,08,900.00 | | |
| Stock-in-hand | Current Assets | 13,23,331.73 | | |
| Sundry Debtors | Current Assets | 62,66,734.18 | | |
| Advance Tax | Current Assets | 2,50,000.00 | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| | | | | |
| TOTAL | | 3,16,47,171.92 | | |

Figure 3.6 Sub Groups

Columnar Report

The reports in which the number of columns added or deleted as per the user inputs are referred to as column based reports. There are four types of column based reports in Tally, namely Multi-Column Reports, Auto-Column Reports and Automatic Auto-Column Reports, Columnar Report. Using Column Report & Repeat attribute at the Report By using the Column Report & Repeat attribute at the Report, "New Column", "Alter Column" and "Delete Column" buttons will be automatically added to 'MulCol TrialBalance' Report.

```
[Report: MulCol Trial Balance]
  ColumnReport: MyMultiColumns
  Repeat       : SVCcurrentCompany, SVFromDate, SVToDate
```

In the above code attribute **Column Report** of the Report definition, facilitates the creation of multi-column reports.

The attribute Column Report is associated with a variable. The variable is specified in the **Repeat** attribute of Report definition. Both attributes must be specified in the Report definition to create a multi-column report.

| Particulars | ABC Company Ltd 1-Apr-2009 to 31-Mar-2010 Closing Balance | |
|---------------------|---|-----------------------|
| | Debit | Credit |
| Capital Account | | 55,00,000.00 |
| Current Assets | 3,17,21,984.42 | |
| Current Liabilities | | 51,38,956.10 |
| Fixed Assets | 34,50,489.68 | |
| Investments | 5,00,000.00 | |
| Loans (Liability) | | 27,27,116.03 |
| GRAND TOTAL | 3,56,72,474.10 | 1,33,66,072.13 |

Figure 3.7 Multi Column Report

Lesson 4: Customisation – An Insight

Lesson Objectives

On completion of this lesson, you will be able to

- Understand the basic concepts for Customisation
- Understand the different types of Customisation

4.1 Customising Default Screens

4.1.1 User Defined Fields

There may be instances when you need to store additional information in the existing objects. This need has given rise to the concept of User Defined Fields (UDF) which is stored as a part of Tally database. In other words, UDFs store additional information in the existing objects.

Similar to data types used in Fields and Variables, **User Defined Fields (UDF)** can also be of type **String, Amount, Quantity, Rate, Number, Date** and **Logical**. Defining UDFs does not serve the purpose unless it is associated with one or more internal objects. When UDF is created in an already existing report, it is always attached to the object to which this report is associated.

Example:

```
[System: UDF]
MyUDF : String : 20001
```

The above code snippet is used to store the user input in the UDF MyUDF using the attribute storage. UDFs should be defined under the section **[System: UDF]**.

MyUDF is the Name of UDF which identifies the UDF it describes the purpose for which it is created. **String** is the datatype which deals with any of the Tally data type. 20001 is the UDF index number which can be any number between 1 and 59999.



Index Number can be any number between 1 and 59999. The number falling between 1 to 9999 and 20001 to 59999 are opened for customisation and 10000 to 20000 is allotted for Common development in TSPL. The user can create 59999 UDFs of each data type.

```
[Field: NewField]
  Use      : NameField
  Storage : MyUDF
```

In the above code snippet, MyUDF is the Name of UDF and the attribute storage allows the user to store the value in the context of current object.

To retrieve the value of MyUDF from an object use the following field definition

```
[Field: MyField]
  Use      : Namefield
  Set as   : $MyUDF
```

UDFs are classified into two types, which are as follows:

- Simple UDF
- Complex/Compound/Aggregate UDF

Simple UDF

A Simple UDF can store one or more values of single data type only. When a UDF is used for storage, it stores the value in the context of object associated at Line or Report Level, by default. Only one value is stored in this case.

Example:

;;A company wants to store their company location which is not available in default Tally. Hence you are adding the same.

```
[#Part : Basic Company]
  Add : Line : At End : CustLocation

[Line : CustLocation]
  Field : Medium Prompt, CustLocation
  Local : Field : Medium Prompt : Set as : "Company Location"
```

[Field : CustLocation]

Use : Name Field
 Storage : CompanyLocation
 Table : CompanyLoc
 Show Table : Always

[Table : CompanyLoc]

Title : "Company Location"
 List : "Bangalore", "Mumbai", "Kolkata", "Chennai"

[System : UDF]

CompanyLocation : String : 30001

In the TDL code, an existing Part **Basic Company** has been modified to add a new Line **CustLocation** which in turn has a Field **CustLocation** that allows the user to enter and store the location of the company in a UDF **CompanyLocation**. You need not associate any object explicitly, since by default this Part inherits the Object, **Company** from the Report.

The Company Alteration screen is displayed as shown.

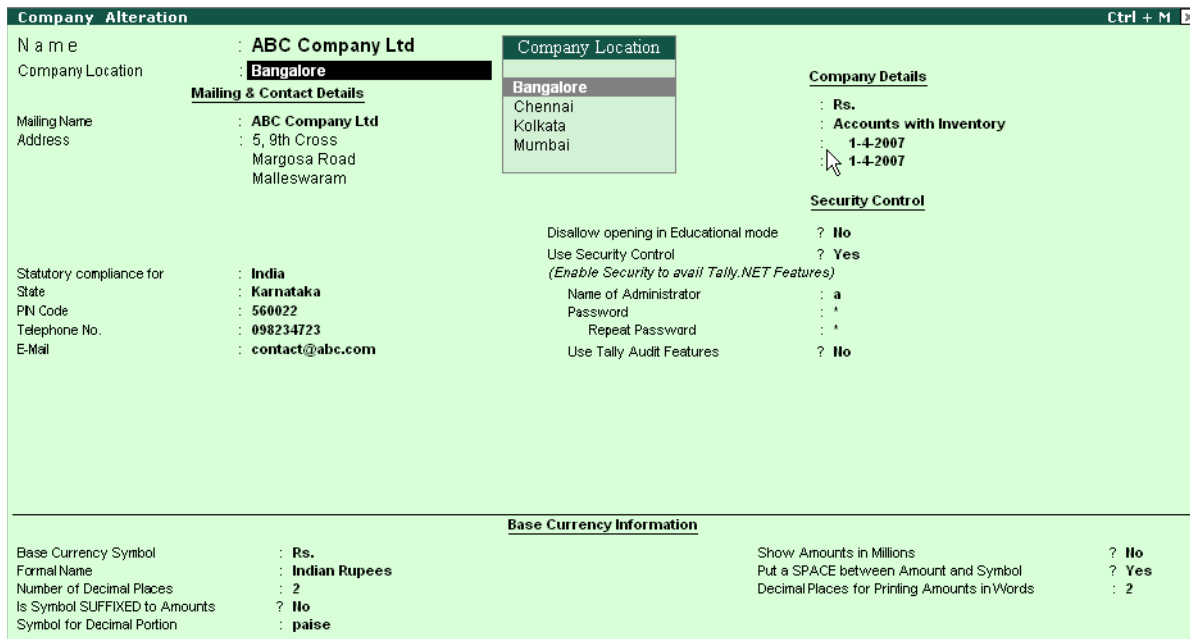


Figure 4.1 Simple UDF Screen

You have added a new input, **Company Location**, to store additional information, which is a UDF of type **String**. A table has been created to restrict the user input to a particular list.

Consider the following scenario when multiple values of the same data type are to be stored as a part of the UDF.

Example:

:: A company wants to create and store the details of vehicles belongs to it.

```
[#Menu: Gateway Of Tally]
  Key Item : Company Vehicles : C : Alter : CompanyVehicles

[Report: CompanyVehicles]
  Form      : CmpVeh
  Object    : Company
```

Here we are associating the Object "Company" at Report level

```
[Form: CmpVeh]
  Part          : CmpVeh
  Width         : 25% Page
  Height        : 60% Page
  Vertical Align : Centre

[Part: CmpVeh]
  Line          : CompTitle, CmpVeh
  Repeat        : CmpVeh : Vehicle
  Break On      : $$IsEmpty:$Vehicle
  Scroll        : Vertical
```

When multiple values of the same data type are to be stored, then the Repeat attribute of Part is used. The field of the line uses the same UDF name in the Storage attribute. In the Repeat attribute **CompVeh** is the name of the line to be repeated and Vehicle is the name of the UDF to store multiple values.

The example explained in the section "UDF to store single value" can be modified to store multiple values of string type.

In this scenario, multiple values of type string can be stored under the object **Company**.

```
[Line: CompTitle]
  Fields        : Form Subtitle
  Local         : Field : Form SubTitle : Set as : "Company Vehicles"
  Space Bottom : 1
```

[Line: CompVeh]

Field : CSrNo, CVeh

[Field: CSrNo]

Use : Number Field

Width : 4

Set as : \$\$Line

Skip : Yes

[Field: CVeh]

Use : Name Field

Storage : Vehicle

Unique : Yes

[System: UDF]

Vehicle : String: 700

The following screen represents a repeat UDF that stores the details pertaining to Company Vehicles.

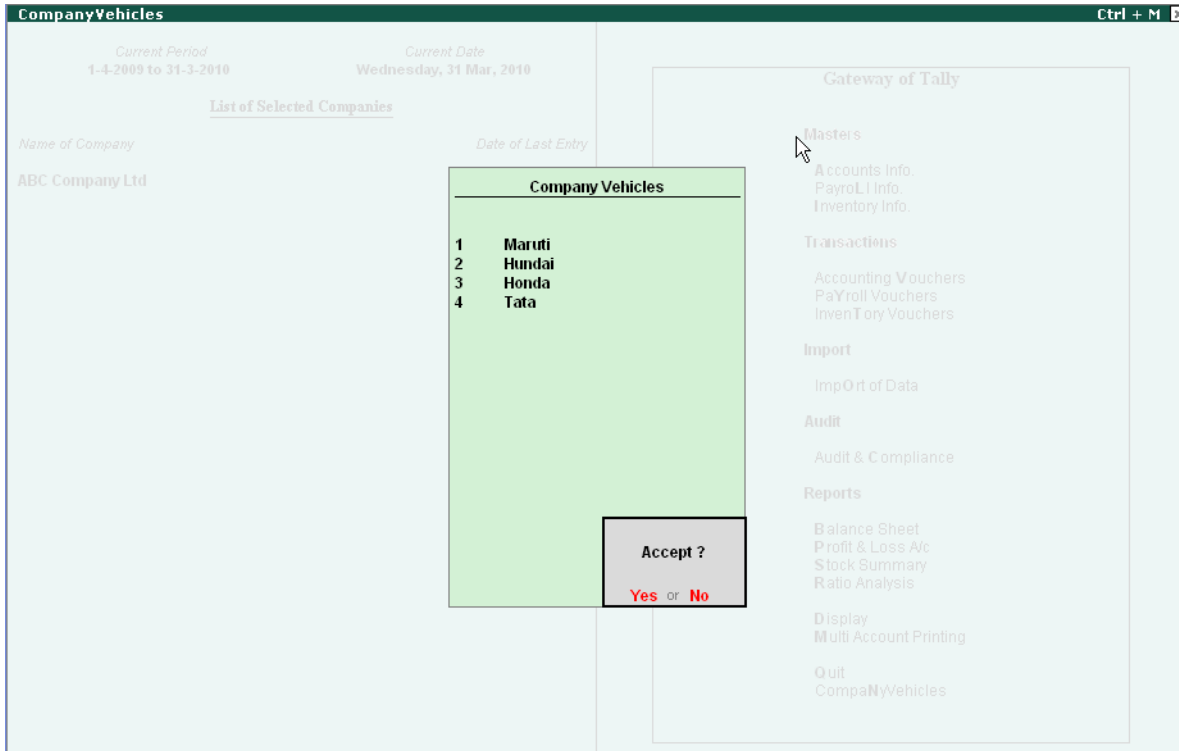


Figure 4.2 Store Company Vehicle

Example:

To display the Company Vehicles stored using the previous code in the Sales Voucher Entry
 [Collection: CMP Vehicles]

```
Type           : Vehicle: Company
Childof        : ##SVCURRENTCOMPANY
Format         : $Vehicle, 20
Title          : "Company Vehicles"
```

```
[#Part: EI DelNoteInfo]
```

```
Add           : Option           : Veh EI DelNoteInfo : @@ISSALES
```

```
[!Part: Veh EI DelNoteInfo]
```

```
Add           : Line : EI Vehicles Det
```

```
Height         : 4
```

[Line: EI Vehicles Det]

Field : Medium Prompt, EI Vehicles Det
Local : Field: Medium Prompt :Set as : "Vehicle : "

[Field: EI Vehicles Det]

Use : Short Name Field
Table : CMP Vehicles, Not Applicable
Show Table : Always
Storage : VCHVehicle

A popup table is displayed when the cursor is placed in the field 'EI Vehicles Det'. The Table contains values stored in the UDF which are Not Applicable as a list.

[System: UDF]

VCHVehicle : String :20001

The following screen shows a new input field added to store the vehicle details in the Invoice. This field should retrieve the table from the Vehicle list.

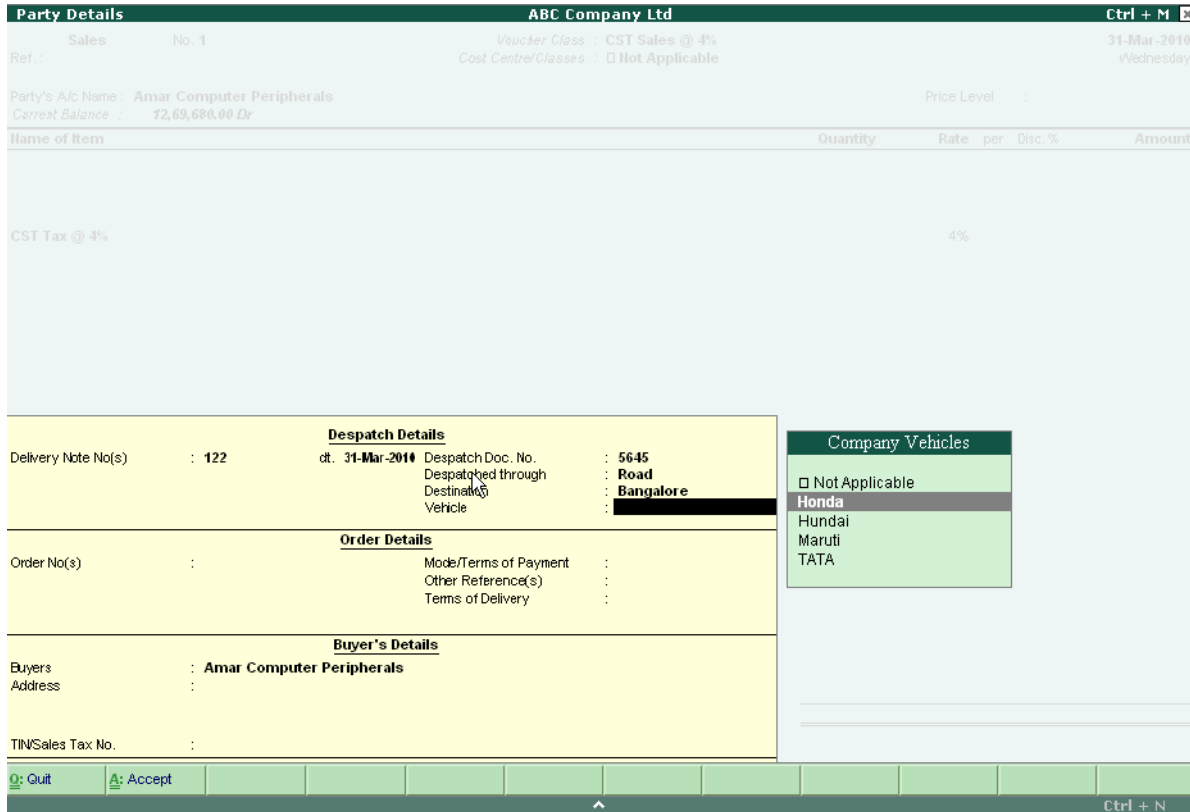


Figure 4.3 Company Vehicle Diapatch

Aggregate UDF

Aggregate UDFs are very useful for storing multiple values and repeated values. An aggregate UDF is a combination of different types of UDFs. Aggregate UDFs can be used to store user data in a tabular format attached to any internal object and can be used as a collection of UDFs. Consider the following example to explain Aggregate UDF.

Example:

A company wants to create and store multiple details of company vehicles. The details required are: Vehicle Number, Brand, Year of Mfg., Purchase Cost, Type of Vehicle, Currently in Service, Sold On date and Sold for Amount

```
;; Report to accept vehicle details
[Report: Company Vehicles]
    Form      : Company Vehicles
    Object    : Company
```

```
;; Definition of Aggregate UDF
```

```
[System : UDF]
```

```
Company Vehicles      : Aggregate : 1000
VVehicle Number      : String      : 1000
VBrand                : String      : 1001
VYear of Mfg          : Number      : 1000
VPurchase Cost        : Amount      : 1000
VType of Vehicle      : String      : 1002
VCurrently in Service : Logical     : 1000
VSold On date         : Date        : 1000
VSold for             : Amount      : 1001
```

```
[Collection: CMP Vehicles]
```

```
Type                : Company Vehicles : Company
Childof             : ##SVCcurrentCompany
Format              : $VVehicleNumber, 20
Format              : $VBrand, 10
Filter              : InServiceFormula
Title               : $$LocaleString:"Company Vehicles"
```

```
[System: Formula]
```

```
InServiceFormula : $VCurrentlyinService
```

The code creates an aggregate UDF to accommodate the details like, Vehicle Number, Brand, Year of Mfg., Purchase Cost, Type of Vehicle, Currently in Service, Sold On date and Sold for.



Aggregate UDF definition does not associate each component field with the aggregate UDF. The association will take place only when you repeat a Line over aggregate UDF and within that Line you have fields which stores value into the component UDFs.

4.1.2 Voucher Customization

A voucher is a primary document that contains all the information regarding a transaction. To begin with, it is necessary to understand the classification of Vouchers and their structure.

Classification of Vouchers

For every transaction in Tally.ERP 9, you can make use of an appropriate voucher to enter all the required details. Broadly vouchers are classified into three types:

- **Accounting Vouchers** - It record transaction which requires only accounting details. Eg., Receipt, Payment, Contra and Journal
- **Inventory Vouchers** - It record transaction which require details pertaining only to Inventory. Eg., Stock Journal and Physical Stock Vouchers
- **Accounting-cum-Inventory Vouchers** - These types of vouchers are transactions which contain details pertaining to both accounts as well as inventory. Eg, Purchase Order, Receipt Note, Rejection In, Debit Note, Purchase, Sales Order, Delivery Note, Rejection Out, Credit Note, Sales etc.

Structure of a Voucher Object

Voucher Objects store two types of information, Base Information and Actual Entries.

Base Information consists of base methods like Voucher Number, Date, Reference, Narration and so on, which are common to all the voucher types.

Actual Entries are the entries pertaining to Accounts and Inventory.

The hierarchy of the **Voucher Objects** is as shown.

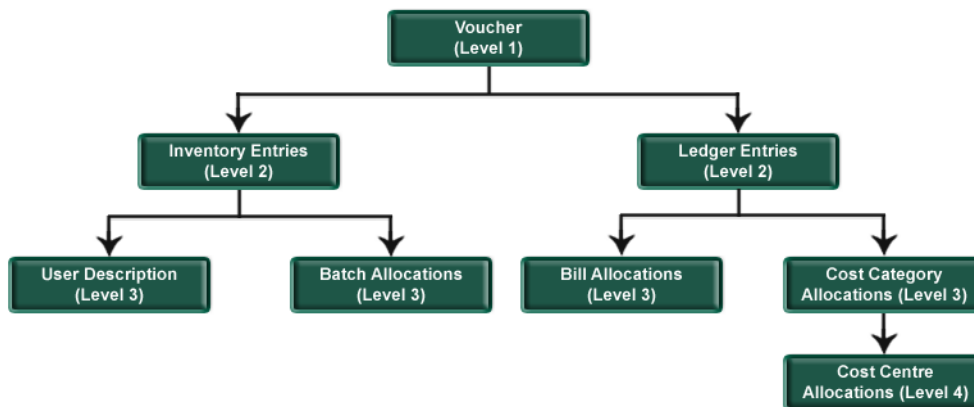


Figure 4.4 Heirarchy of Voucher Objects

The base entries of the Voucher are Date, Voucher Type, Voucher Number, etc.

At the first level are two basic collections namely, Ledger Entries and Inventory Entries. Each Ledger Entry Object has its own base Methods like LedgerName, Amount, Bill Allocation Collection and Cost Category Allocation Collection. Each Cost Category Allocation Object in turn, contains its own Methods, which are Name, Amount and a Cost Centre Allocation Collection.

4.1.3 Customisation - Case studies

A user usually enters transactions in a voucher and prints it in the default format provided. However, there may be instances, when the user would want to print in a format other than the

default one provided in Tally. In such circumstances, user may have to get it customised according to the company needs.

When there is a requirement for customisation, you need to adhere to the following steps:

1. Analyse the Format required by the company to judge whether
 - The requirement can be met with the default format with some minor changes.
 - OR
 - A whole new format needs to be designed.
2. Check whether any additional input fields are required. If required, add the appropriate UDFs at relevant places.
3. Identify the definitions that need to be altered to suit the user requirements.

Voucher Customisation - Case studies

Let us consider the following examples to understand the Voucher Customisation.

Case 1

A Company **ABC Company Ltd** needs Cheque No., Date and Bank Name in Payment/ Receipt Voucher and Printed Receipt. Also there should be an option whether to print the Cheque details or not.

Solution:

Step 1: - Add additional fields to capture Bank Name, Cheque Number and Cheque Date. For this UDFs following UDF are created

```
[System: UDF]
BankName      : String      : 1000
NarrWOCh      : String      : 1001
ChequeNumber  : Number      : 1000
ChqDate       : Date        : 1000
```

Above UDFs are used in the existing Part VCH Narration.

;; Modify the Narration Part to add the details

```
[#Part: VCH Narration]
Add: Option: BankDet VCH Narration      :@@IsPayment OR @@IsReceipt
Add: Option: BankDet VCH NarrationRcpt : @@ReceiptAfterSave
```

The Screen of the Receipt Voucher on entering the details looks like this:

| Accounting Voucher Alteration (Secondary) | | ABC Company Ltd | | Ctrl + M |
|---|-----|---|----------------------------|-----------------------|
| Receipt No. 211 | | Cost Centre/Classes : Not Applicable | | 31-Mar-2008 Monday |
| Account : HDFC Bank | | | | |
| Cur Bal : 1,39,229.82 Cr | | | | |
| Particulars | | | | Amount |
| Modern Advertisers | | | | 7,303.40 |
| Cur Bal : 0.00 Cr | | | | |
| Agst Ref | 111 | 7,303.40 Cr | Output ST - Advt. Services | |
| Name on Receipt : Modern Advertisers | | | | |
| Cheque Number : 11384 Dated : 31-Mar-2008 | | | | |
| Vide Bank : Canara Bank | | | | |
| Remark : Full Amount is being received | | | | |
| Narration : | | | | |
| 11384 | | | | 7,303.40 |

Figure 4.5 Alteration of Receipt Voucher

Step 2:- Configuration screen of Receipt & Payment is altered to add a new option. For this existing Parts Payment Print Config & Receipt Print Config are altered

:: Payment Config Changes

```
[#Part: Payment Print Config]
```

```
Add: Lines : Before: PPRVchNarr : PPR ChqDetails
```

:: Receipt Config Changes

```
[#Part: Receipt Print Config]
```

```
Add: Lines : After: PPRWithCost: PPR ChqDetails
```

Step 3:- Existing **Field PPR Narr & Part PPRBottomDetails** is altered to get required Receipt/ Payment Voucher

```
[#Field: PPR Narr]
```

```
Option : PPR Narr Rct Pymt
```

```
[#Part: PPRBottomDetails]
```

```
Option : PPRBottomDetails Rct Pymt : (@@IsPayment OR @@IsReceipt)
```

```
AND ##PPRChqInfo
```

The Print out of Customised Receipt Voucher looks like this:

ABC Company Ltd
 5, 9th Cross
 Margosa Road
 Mallewaram
Receipt Voucher

No. : 211 Dated : 31-Mar-2008

| Particulars | Amount |
|--|----------|
| Account : | |
| Modern Advertisers | 7,303.40 |
| Agst Ref 111 7,303.40 Cr Output ST - Adv. Services | |
| Cheque Number and Date : 11384 dt 31-Mar-2008 Through : HDFC Bank On Account of : Full Amount is being received Amount (in words) : Rs. Seven Thousand Three Hundred Three and Forty paise Only | |
| | 7,303.40 |

Authorised Signatory

Figure 4.6 Print Preview of Receipt Voucher

Step 4:- Existing Field PRCT Thru is altered to get required Receipt/Payment Voucher

The Print out of Customised Receipt looks like this:

[#Field: PRCT Thru]

Option : PRCT Thru Rct Pymt : @@IsReceipt

No.: 211

Dated 31-Mar-2008

ABC Company Ltd
5, 9th Cross
Margosa Road
Malleswaram

RECEIPT*Recd with thanks from* : **Modern Advertisers***The sum of* : **Rs. Seven Thousand Three Hundred Three and
Forty paise Only***By* : **Cheque Number 11384 dated 31-Mar-2008 drawn on Yes Bank***Remarks* : **Full Amount is being received****Rs. 7,303.40**

Authorised Signatory

Figure 4.7 Print Preview of Receipt Voucher

4.2 Invoice Customisation

Invoice customisation can be broadly classified into following categories based on the requirement.

- Invoice Customization - User defined format
- Invoice Customization - Modifications to default format

4.2.1 Invoice Customization - User defined format

In this category, totally new Invoice format needs to be developed. After developing a new Invoice format, this can be enabled in following two different ways.

1. Replacing the existing format with new one
2. Adding new format along with default format

- **Replacing the existing format with new one**

By default, there are two basic formats provided for Commercial Invoice Printing

1. Normal Invoice i.e., Comprehensive Invoice
2. Simple Invoice i.e., Simple Printed Invoice

Comprehensive Invoice and Simple Printed Invoice are two optional forms which are executed based on satisfying a given condition. The default option available for print is the **Comprehensive Invoice**.

Simple Invoice is printed if **Print in Simple Format** is set to **Yes** in **F12 Configuration**. Comprehensive Invoice is printed only if the user opts for **Neat Format** mode of printing and above Option set to No.

Case 1

ABC Company Ltd requires Sales Invoice in the following format for both Normal Invoice as well as Simple Invoice.



ABC Company Ltd

Voucher Date : 1-8-2008

Voucher No : 2

Party Name : Universal Systems

| Sr No | Name | Billed Qty | Rate | Amount |
|---------------|----------------------------|------------|-----------|------------------|
| 1 | Assembled PIV | 20 | 22,000.00 | 440000.00 |
| | Transportation & Packaging | | | 2500.00 |
| | CST Tax @ 4% | | 4% | 17600.00 |
| Totals | | 20 | | 460100.00 |

Amount in words : Rs. Four Lakh Sixty Thousand One Hundred Only

For ABC Company Ltd

Authorised Signatory

Figure 4.8 Invoice Customisation

Solution:

Step 1:-Default forms Comprehensive Invoice & Simple Printed Invoice are modified with an optional Form.

```
[#Form: Comprehensive Invoice]
    Add          : Option: My Invoice: @@IsSales

[#Form: Simple Printed Invoice]
    Add          : Option: My Invoice: @@IsSales
```

Step 2:-The Parts and Page Breaks of the default Form are deleted and new Parts are added. To begin with, the Invoice is classified into three parts: Top Part, Body Part and Bottom Part. These Parts can be further divided into any number of Parts according to the user's requirement.

```
[!Form: My Invoice]
    Delete       : Parts
    Delete       : Bottom Parts
    Delete       : PageBreak
    Space Top    : 0
    Space Bottom : 0
    Space Left   : 0
    Space Right  : 0
    Add: Part    : My Invoice Top Part
    Add: Part    : My Invoice Body Part
    Add: Bottom Part: My Invoice Bottom Part
```

- **Adding new format along with default format**

Alternatively, you can also create a new format of invoice by modifying the existing Form **Sales Color** by replacing its default print report code.

This way of Customisation begins with the following code snippet:

```
[#Form: Sales Color]
    Add          : Print: Sales Invoice

[Report : Sales Invoice]
    Form         : Sales Invoice
    Object       : Voucher
```

In this code snippet, the default **Print Report** is deleted, the Report **Sales Invoice** is added and the Object Voucher is associated to it. However, in the previous example, it was not necessary to associate the Voucher Object, since it was already associated in the default Report, **Printed Invoice**.

Case 1

ABC Company Ltd requires Sales Invoice in addition to the default sale invoice.

| INVOICE | | | | |
|---|---------------|---|------------|--|
| Billing Name & Address Universal Systems | | Shipping Address | | Inv No : 2 Inv Dt : 1-Aug-2008 Terms of Delivery : Due Dt : 30-Oct-2008 Shipped Dt : Ship Via : |
| SI. No. | Item Name | Quantity | Rate | Amount |
| 1 | Assembled PIV | 20 Nos | 220,000.00 | 4,40,000.00 |
| Sub Total | | | | 4,40,000.00 |
| Transportation & Packaging | | | | 2,500.00 |
| CST Tax @ 4% | | | | 17,600.00 |
| Net Amount | | | | 4,60,100.00 |
| Address 5, 9th Cross Margosa Road Malleswaram | | Phone : 098234723 Fax : Email : contact@abc.com | | for ABC Company Ltd |

Figure 4.9 Invoice Customisation

4.2.2 Invoice Customization - Modifications to default format

There may be a requirement in Invoice customisation which is similar to the default Tally format with some minor changes. In such cases, one can just alter the default definitions as required.

Case 1

A Company ABC Company Ltd requires an Invoice to have Terms and Conditions as shown below:

| Tax Invoice | | | | | | | | | | | |
|---|-----------------------------|---------------------|----------------------------|--|-----------|-----|-----------------------------|-------------|--|--|--|
| ABC Company Ltd 5, 9th Cross Margosa Road Malleswaram E-mail : contact@abc.com | | | | Invoice No. | 2 | | Dated | 1-Aug-2008 | | | |
| | | | | Delivery Note | | | Mode/Terms of Payment | 90 Days | | | |
| | | | | Supplier's Ref. | | | Other Reference(s) | | | | |
| Buyer Universal Systems | | | | Buyer's Order No. | | | Dated | | | | |
| | | | | Despatch Document No. | | | Dated | | | | |
| | | | | Despatched through | | | Destination | | | | |
| | | | | Vessel/Flight No. | | | Place of Receipt by Shipper | | | | |
| | | | | City/Port of Loading | | | City/Port of Discharge | | | | |
| | | | | Terms of Delivery | | | | | | | |
| Sl No. | Marks & Nos./ Container No. | No. & Kind of Pkgs. | Description of Goods | Quantity | Rate | per | Dist. % | Amount | | | |
| 1 | | | Assembled PIV | 20 Nos | 22,000.00 | Nos | | 4,40,000.00 | | | |
| | | | Transportation & Packaging | | | | | 2,500.00 | | | |
| | | | CST Tax @ 4% | | | 4 % | | 17,600.00 | | | |
| Total | | | | 20 Nos | | | | 4,60,100.00 | | | |
| Amount Chargeable (in words) | | | | | | | | E. & O.E | | | |
| Rs. Four Lakh Sixty Thousand One Hundred Only | | | | Terms & Conditions : | | | | | | | |
| | | | | 1. Training : 9 hours | | | | | | | |
| | | | | 2. Tally Support : 3 months without any additional charges | | | | | | | |
| Company's VAT TIN : 23424872389 | | | | | | | | | | | |
| Company's CST No. : 234234234 | | | | | | | | | | | |
| Company's Service Tax No. : 234234 | | | | | | | | | | | |
| Company's PAN : EENMM16789 | | | | | | | | | | | |
| Declaration | | | | for ABC Company Ltd | | | | | | | |
| We declare that this invoice shows the actual price of the goods described and that all particulars are true and correct. | | | | Authorised Signatory | | | | | | | |

This is a Computer Generated Invoice

Figure 4.10 Invoice Customisation

Solution:

Step 1:- Default configuration Part IPCFG Right is altered to add ne option.

[#Part: IPCFG Right]

Add : Lines : GlobalWithTerms

Step 2:- Default Part EXPINV ExciseDetails is altered to cater the requirement.

```
[#Part: EXPINV ExciseDetails]
  Delete : Lines : EXPINV ExciseRange, EXPINV ExciseRangeAddr, +
          EXPINV ExciseDiv, EXPINV ExciseDivAddr, +
          EXPINVExciseSerial,EXPINVInvoiceTime,EXPINV RemovalTime
  Add : Lines : EXPINV SubTitle, EXPINV ExciseDetails
  Repeat: EXPINV ExciseDetails : Global Terms
  Local : Field: EXPINV SubTitle : Info : "Terms & Conditions :"
  Local : Field: EXPINV SubTitle : Border : Thin Bottom
  Local : Line : EXPINV SubTitle : Space Bottom : 1
  Invisible : NOT @@IsInvoice OR NOT ##ShowWithTerms
```

Case 2

Sorting Inventory Entries as per user requirement

Solution:

The inventory entries of an invoice are printed in the order in which they are entered. This order can be changed as per user requirement. The sorting can be done in either the ascending or descending order of the item name, stock group, and stock category, units of measure, rate, and value and so on. To denote the descending order, attach '-' sign to it.

To change the order of the default invoice:

- Define a Collection of inventory entries in the desired sorted order

```
[Collection : Sorted Inventory Entries]
```

```
Type : Inventory Entries : Voucher
```

```
Sort : Default : -$Parent:StockItem:$StockItemName, $StockItemName
```

- Note the Part in which the repeat Line of Inventory entries is mentioned in the DefTDL and Change this Part to repeat the Line with the new Collection defined