



POWER OF SIMPLICITY

Tally.ERP 9 – Integration Capabilities

The information contained in this document represents the current view of Tally Solutions Pvt. Ltd., ('Tally' in short) on the topics discussed as of the date of publication. 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.

This document is for informational purposes only. TALLY MAKES NO WARRANTIES, EXPRESS OR IMPLIED, IN THIS DOCUMENT.

Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright, no part of this document may be reproduced, stored in, or introduced into a retrieval system, or transmitted in any form, 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, Tally 9, Tally9, Tally.ERP, Tally.ERP 9, Shoper, Shoper 9, Shoper POS, Shoper HO, Shoper 9 POS, Shoper 9 HO, TallyDeveloper, Tally Developer, Tally.Developer 9, Tally.NET, Tally Development Environment, Tally Extender, Tally Integrator, Tally Integrated Network, Tally Service Partner, TallyAcademy & Power of Simplicity are 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: Tally.ERP 9 – Integration Capabilities/1.0/July 2009

Contents

Lesson1: Integration –The Overall Perspective	1
0.1 Introduction	1
0.2 Need and Benefits of Integration	1
0.3 Tally Interfaces – For Integration	2
0.3.1 Tally ODBC Interface (Read Only)	2
0.3.2 Tally XML Interface (Read and Write)	3
0.4 XML Messaging Formats	3
0.4.1 Template used for XML Message Format	4
0.4.2 Request Template	4
0.4.3 Response Template	5
0.4.4 Generic Failure Format	6
0.5 Components of Request / Response	7
0.5.1 Header Information	7
0.5.2 Body Information	8
0.6 Significance of all Tags	8
0.6.1 Header Tags	8
0.6.2 Body Tags	11
0.7 Case Study – Using the above XML Request/Response Formats	17
0.7.1 Export	17
0.7.2 Import	33
0.7.3 Execute	39
Lesson2: Integration Using XML Interface	41
0.8 Tally.ERP 9 as a Server – Using External application as Front End	42
0.8.1 Case Study I – Importing Masters from Excel to Tally.ERP 9	42
0.8.2 Case Study II - Creation and Alteration of Vouchers through VB	47
0.8.3 Case Study III – Exporting Ledger Masters from Tally.ERP 9 to External Application	59
0.9 Tally.ERP 9 as a Client – Tally as a Front end for Web Services	62
0.9.1 Introduction	62
0.9.2 Collection attribute – Remote URL	64
0.9.3 Collection attribute – XML Object Path	65
0.9.4 Collection attribute – XML Object	65
0.9.5 Collection attribute – Remote Request	66
0.9.6 Action – HTTP POST	68
0.9.7 Event – On Form Accept	69
0.9.8 Event – On Focus	70
0.10 Collection Capability to Accept File as a Data Source	70
0.10.1 The Collection attribute – Data Source	70
0.11 Case Study	71

Lesson 3: Integration using ODBC Interface	79
0.12 Tally.ERP 9 – ODBC Interface	79
0.13 Tally.ERP 9 as a Client – Retrieving Data from External Database	81
0.13.1 TDL Collection to gather data from MS Access	82
0.13.2 TDL Collection to gather data from MS Excel	84
0.14 Tally.ERP 9 as a Server – Retrieving Data from Tally DB using an External Application	84
0.14.1 Retrieving Data Using Tables	85
0.14.2 Retrieving Data By Calling an SQL Procedure	94
0.14.3 Collection Attribute – SQLParms	95
0.14.4 Collection Attribute – SQLValues	95

Lesson1: Integration –The Overall Perspective

Lesson Objectives

On the completion of this chapter you will be able to

- ❑ Understand the need and benefits of Integration
- ❑ Understand the different Tally Interfaces used for Integration
- ❑ Understand the XML messaging format using XML Interface

1.1 Introduction

Large and medium sized businesses use disparate applications to run their business and one of the major areas that need to converge amongst these applications is the Accounting, Financial and Inventory information. Tally being the default accounting, Inventory and Statutory Compliance software used by enterprises in these segments. Therefore the need arises to discuss on the Integration Capabilities of Tally.

Integration Solutions are designed to ensure that the existing investments in Software (ERP, Legacy and other Enterprise systems) remain intact by seamlessly integrating information with new systems, technologies and custom applications within the enterprise, as well as with companies with whom the business deals with.

1.2 Need and Benefits of Integration

To meet the challenges of the new business environment, information systems need to communicate with each other as seamlessly as possible, provide right-time visibility of transactions across the entire enterprise and be flexible enough to accommodate the changing structure of the business. When more and more information needs to be shared across traditional business boundaries, the way you integrate your systems and processes is rapidly becoming one of the most important priorities in business today.

The following figure gives a complete perspective on the overall Integration Capabilities of Tally.ERP 9

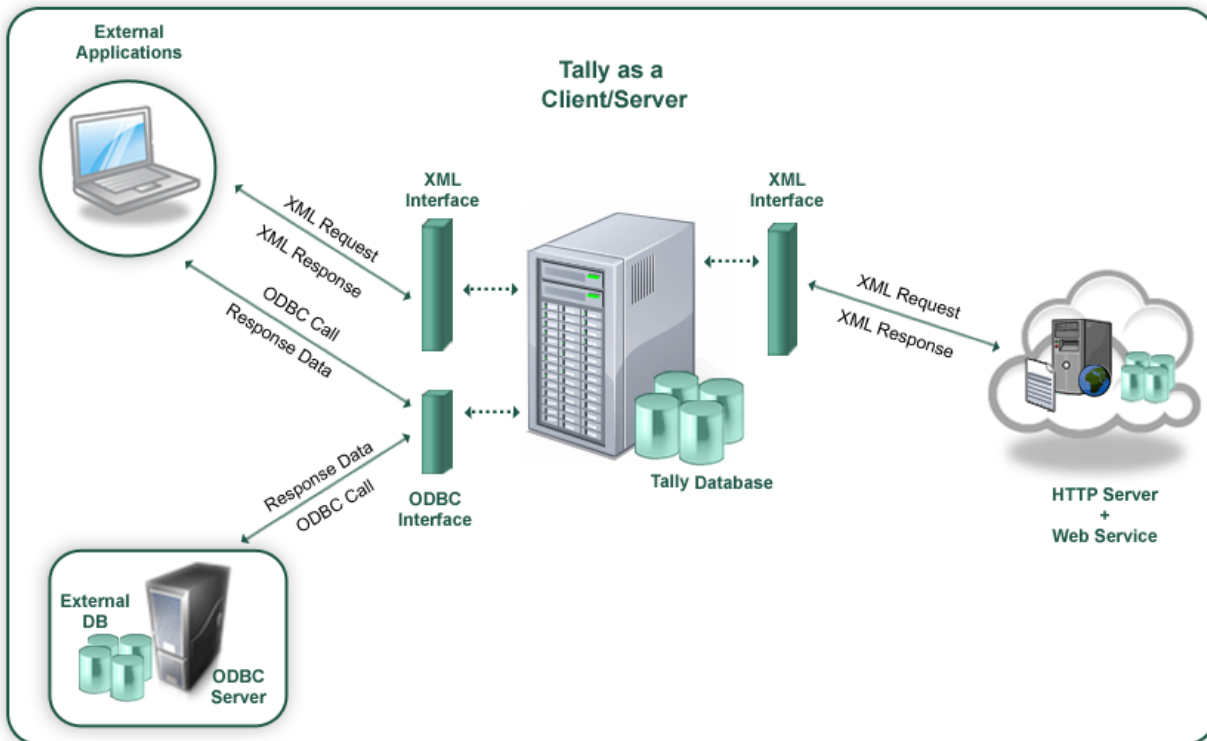


Figure1.1 Tally.ERP 9 – Integration Capabilities

1.3 Tally Interfaces – For Integration

Tally communicates with the external world mainly using two Interfaces.

- Tally ODBC Interface (Read Only)
- Tally XML Interface (Read and Write)

1.3.1 Tally ODBC Interface (Read Only)

ODBC (Open Database Connectivity) makes it possible to access data from any application, regardless of which Database Management System (DBMS) is handling the data. ODBC manages this by inserting a middle layer, called a database driver between an application and the DBMS. The purpose of this layer is to translate the application's database queries into commands that the DBMS can understand. For this to function, both the application and the DBMS must be ODBC compliant i.e., the application must be capable of issuing ODBC commands and the DBMS must be capable of responding to them.

Tally provides the ODBC Interface which makes it possible for applications to talk to Tally Database. By using this interface, external applications will be able to retrieve data from Tally. Tally acts as a Server delivering Data to external applications.

Using the ODBC Interface, Tally.ERP 9 can make ODBC calls to an External Database and retrieve data from them. In such a case Tally acts as a Client to pull Data from disparate Data Sources. This data can be consumed in Tally as per requirement.

The usage and techniques for the same will be discussed in Lesson 3.

1.3.2 Tally XML Interface (Read and Write)

XML (Extensible Markup Language) is the standard for information exchange with external systems. Tally.ERP 9 supports standardized message formats for Request/Response. Tally.ERP 9 can communicate with any environment capable of sending and receiving XML over HTTP.

Tally can act as an HTTP Server capable of receiving an XML Request and responding with an XML Response. The entire Tally Data can be made available to the requesting application. It is also possible for the application to store data into Tally Database.

Using the same interface, Tally has the capability to interact with a Web Service delivering Data over HTTP. In this scenario, Tally behaves as a client retrieving and storing data into an external database. The Web Service capable of handling Tally Request/Response serves as a layer between Tally and External Database.

In this Lesson we will be discussing the XML Messaging Formats supported by Tally. The application and usage will be discussed in detail in Lesson 2.

1.4 XML Messaging Formats

A message format is an encoded spatial or time-sequential arrangement of the parts of a message that is recorded in or on a data storage medium.

XML Messaging format is specified for exchanging structured information in the implementation of Web Services in computer networks. An XML interface can form the foundation layer of a web services protocol stack, providing a basic messaging framework upon which web services can be built.

XML standard format has lots of potential as a data representation and messaging mechanism. Data representation typically involves translating the data from a local format into XML, and then back into the same format or even a completely different one, on the other end of a connection.

In Tally.ERP 9, XML messaging format is used for the purpose of integration. Here Tally.ERP 9 uses XML format for communication with external applications including other instances of Tally.ERP 9. The data exchange happens by way of Request / Response. Tally.ERP 9 identifies certain tags for Request and sends a Response accordingly based on the Request.

All Requests and Responses are used in Tally.NET Messages contain custom HTTP headers to identify the requests that needs to be processed or forwarded.



An XML fragment is everything from the start tag to the end tag. A fragment can contain other fragments, simple text or a mixture of both. Fragments can also have attributes. XML documents do not carry information about how to display the data. Since XML tags are "invented" by the author of the XML document, browsers do not know if a tag like <table> describes an HTML table or a dining table. Without any information about how to display the data, most browsers will just display the XML document as it is.

1.4.1 Template used for XML Message Format

Tally.ERP 9 follows the XML interface for exchanging data with other systems or with other Tally.ERP 9 instances. This XML interface specifies the following format for communication.

```
<ENVELOPE>
  <HEADER> . . . </HEADER>
  <BODY> . . . </BODY>
</ENVELOPE>
```

1.4.2 Request Template

The XML structure used for requesting messages is as follows:

```
<ENVELOPE>
  <HEADER>
    <VERSION>Version Number</VERSION>
    <TALLYREQUEST>Request Type</TALLYREQUEST>
    <TYPE>Information Type</TYPE>
    <SUBTYPE>Sub Type</SUBTYPE>
    <ID >Identifier</ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        Static Variables Specification
      </STATICVARIABLES>
```



```

    <REPEATVARIABLES>
        Repeat Variables Specification
    </REPEATVARIABLES>
    <FETCHLIST>
        Fetch Specification
    </FETCHLIST>
    <FUNCPARAMLIST>
        Parameter Specification in the case of function type
    </FUNCPARAMLIST>
    <TDL>
        TDL Information
    </TDL>
</DESC>
    <DATA>
        Data (if applicable)
    </DATA>
</BODY>
</ENVELOPE>

```

1.4.3 Response Template

The XML structure used for response is as follows:

```

<ENVELOPE>
    <HEADER>
        <VERSION>Version Number</VERSION>
        <STATUS>-1/0/1</STATUS>
    </HEADER>
    <BODY>
        <DESC>
            <STATICVARIABLES>
                Static Variables Specification
            </STATICVARIABLES>
            <REPEATVARIABLES>
                Repeat Variables Specification
            </REPEATVARIABLES>
            <FETCHLIST>

```

```

        Fetch Specification
    </FETCHLIST>
    <FUNCPARAMLIST>
        Parameter Specification
    </FUNCPARAMLIST>
    <TDL>
        TDL Information
    </TDL>
</DESC>
<DATA>
    Data (if applicable)
</DATA>
</BODY>
</ENVELOPE>
```

1.4.4 Generic Failure Format

In case of a failure, all responses could be made to respond using the following format:

```

<ENVELOPE>
    <HEADER>
        <VERSION>Version</VERSION>
        <STATUS>0</STATUS>
    </HEADER>
    <BODY>
        <DATA>
            <STATUS.LIST>
                <STATUS>
                    <CODE>Code</CODE>
                    <DESC>Description</DESC>
                </STATUS>
                ...
            <STATUS.LIST>
        </DATA>
    </BODY>
</ENVELOPE>
```

Example 1:

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>0</STATUS>
  </HEADER>
  <BODY>
    <DATA>
    </DATA>
  </BODY>
</ENVELOPE>
```

1.5 Components of Request / Response

<ENVELOPE> is the top element of the XML fragment which is representing the message.

Both Request and Response consists of two sections:

- Header
- Body

1.5.1 Header Information

Header section will give all identification information to the recipient such as authentication, transaction management, and payment so on. This section determines how the recipient of the message should process the information. Header information is classified in two ways, one is for Request and the other is for Response. All the information about Request or Response is enclosed with Header Tags.

In case of Request, header information includes mainly four elements which are Version, TallyRequest, Type and ID. Version gives the version of the message format. Second element TallyRequest will identify the type of request as Import or Export in the messaging format. If the value of Tally Request is Import then the type of information would be Data, and the request will be identified by the report name specified in ID. If the value of Tally Request is Export then the type of information would be Data, Collection, Object or Function. The ID specifies the name of Report, Collection, Object or function.

In the case of Response, there are mainly two elements which are Version and Status. Version gives the version of the message format. Status indicates whether the request is success or failure.

1.5.2 Body Information

It exchanges the information intended for the recipient of the message. This section gives the actual details of the message. It is further divided into two sections:

- Description for Request/Response
- Data required for the Request/Response

Description section is used to give the description for message, request or response. Description element mainly includes all types of variable information, storage information, computational information and user defined TDLs. All the description information is enclosed with <DESC> tags.

Data section includes all the data information being transferred. All the data should be enclosed within the <DATA> tags.

1.6 Significance of all Tags

Following are some significant tags that are required for requesting any info from Tally.ERP 9. The tags are divided into two categories:

- Header Tags
- Body Tags

1.6.1 Header Tags

Header tags are enclosed with the tag <HEADER> and </HEADER>. These tag gives all the header information.

<VERSION>

It gives the version of the messaging format. The tag <VERSION> is a mandatory tag which is used in Header tags.

<TALLYREQUEST>

TALLYREQUEST tag specifies the type of request. This tag is vital as it determines the response required. The permissible values for this tag are Import, Export and Execute. This value further determines the range of values required for this tag.

□ Import

Import is specified when we want Tally to import data from the XML fragment to Tally. i.e Tally validates and saves the data. A request is made to Import Data or File which can be specified within the subsequent tags.

□ Export

Export is specified when we want to retrieve data from Tally. A request is made to Export Data, Collection, Object, Function, etc. which is specified within the tag <TYPE>.

□ Execute

Execute is specified when we want to execute some TDLAction in Tally.ERP 9. Here the request is made to Execute TDL Action which can be specified within the <TYPE> tag.

The following table describes the value of <TALLYREQUEST> tag.

Request Type	Action	Comments
Import	Set	Requesting recipient to import the data.
Export	Get	Request for Exporting Data from other end.
Execute	Run	Request for executing the action in Tally

<TYPE>

The <TYPE> tag provides the type of information being requested / responded. The tag TALLYREQUEST determines the value for this tag.

The possible values of <TYPE> tag are as follows:

- ❑ Data - Request for the data or Response of data
- ❑ Collection - Collection Request or Response
- ❑ Object - Object Request or Response
- ❑ Action - Action to be performed by the message recipient
- ❑ Function - Execute the Function Request

<SUBTYPE>

<SUBTYPE> is an optional tag. In cases the <TYPE> tag alone is not sufficient to identify the entity then SUBTYPE tag will be used. For example OBJECT refers to various types of objects such as GROUP, LEDGER, and CURRENCY etc.

<ID>

<ID> tag provides the identification of Request. The TYPE attribute of this tag can be used to identify specific instance of an entity. The attribute Type can be used like the following example:

```
<HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>OBJECT</TYPE>
    <SUBTYPE>Ledger</SUBTYPE>
    <ID TYPE="Name">ABC India Pvt. Ltd. </ID>
</HEADER>
```

TYPE	ID - Value	Qualification
DATA	Name of the Request / Report	Not Applicable
COLLECTION	Name of the Collection	Not Applicable
OBJECT	Object ID or Name of the Object	Object Identifier Attribute
ACTION	Name of the Action to be performed - As of now, only the action Sync is introduced	Not Applicable
FUNCTION	Name of the Function to be executed	Not Applicable

<STATUS>

The <STATUS> tag is applicable only for Response. Value within this tag indicates success or failure of the request made. Possible values are SUCCESS (1), FAILURE (0).

Example for a Failure Response

```

<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>DATA</TYPE>
    <ID>All masters</ID>
  </HEADER>
  <BODY>
    <DESC>
    </DESC>
  </BODY>
</ENVELOPE>

```

Failure Response

```

<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>0</STATUS>
  </HEADER>

```

```
<BODY>
  <DATA>
  </DATA>
</BODY>
</ENVELOPE>
```



At the time of request within <HEADER>...</HEADER>, all tags must have appropriate value. Only <SUBTYPE> is optional and can be used where applicable. In case of Response Header has only two tags i.e., <VERSION> tag to indicate the version of the messaging format and <STATUS> tag to indicate Success / Failure.

1.6.2 Body Tags

As discussed, there are two sections under the tag <BODY>:

- ❑ <DESC>: Description for Request/Response
- ❑ <DATA>: Data required for Request/Response

<DESC>: Description for Request/Response

The description block is used for providing description of Request / Response. Following are the different types of descriptions available inside the description tag:

- ❑ StaticVariables
- ❑ RepeatVariables
- ❑ ComputeList
- ❑ FetchList
- ❑ Function ParamList
- ❑ TDL

<STATICVARIABLES>

The STATICVARIABLES provides all configuration details and all global variable details. All tags inside Staticvariable would be any of the system variables. In the following example SVCCurrent-Company, SVFromDate and SVToDate are the system variables. All elements inside the tag <STATICVARIABLES> would be starting with SV.

```
<DESC>

    <STATICVARIABLES>
        <SVCURRENTCOMPANY>
            ABC Company Ltd
        </SVCURRENTCOMPANY>
        <SVFROMDATE TYPE="Date">1-Apr-2008</SVFROMDATE>
        <SVTODATE TYPE="Date">31-Mar-2009</SVTODATE>
    </STATICVARIABLES>
</DESC>
```

<REPEATVARIABLES>

REPEATVARIABLES is used to specify the details for all the repeated variable information such as Date Range, Block Range etc. All the information related to repeated variables will be enclosed within the tag <REPEATSET>. Here also the value will be enclosed with the system variables. In the following example shows the repeated usage of variables.

```
<REPEATVARIABLES>
    <REPEATSET>
        <SVFROMDATE>1-Apr-2007</SVFROMDATE>
        <SVFROMDATE>1-Oct-2007</SVFROMDATE>
    </REPEATSET>
    <REPEATSET>
        <SVTODATE>30-Sep-2007</SVTODATE>
        <SVTODATE>31-Mar-2008</SVTODATE>
    </REPEATSET>
</REPEATVARIABLES>
```

<FETCHLIST>

FETCHLIST is used to specify the list of storages to be fetched. In case of type object the methods that need to be retrieved within the <FETCHLIST> Tag using <FETCH> Tags.

```
<FETCHLIST>
    <FETCH>TBalClosing</FETCH>
    <FETCH>TBalOpening</FETCH>
    <FETCH>StkClBalance</FETCH>
    <FETCH>StkOpBalance</FETCH>
</FETCHLIST>
```


<FUNCPARAMLIST>

It is used to specify the parameters used for function execution. In the time of using Function as a type with parameters then the parameter list will come under the tag <PARAMLIST>. The structure of PARAMLIST as follows:

```
<FUNCPARAMLIST>
    <PARAM>@@FirstParameter</PARAM>
    <PARAM TYPE="Number">0.10</PARAM>
</FUNCPARAMLIST>
```

<TDL>

It is used to specify the TDL related information. The complete TDL to be executed in order to handle the Request; will be sent within the TDL block. TDL tag is specified, when Report, Collection, Object or Function is to be sent as a request to Tally. Tally application will respond depends on the TDL request. A TDL specification is required only when the TDL required for serving the request does not exist at the Tally end.

The TDL program is sent using TDL tag as per the following structure:

```
<TDL>
    <TDLMESSAGE>
        <REPORT NAME="TDL Report" ISMODIFY="No" ISFIXED="No"
            ISINITIALIZE="No" ISOPTION="No" ISINTERNAL="No">
            <FORMS>First TDL Form</FORMS>
        </REPORT>
        <FORM NAME="First TDL Form" ISMODIFY="No" ISFIXED="No"
            ISINITIALIZE="No" ISOPTION="No" ISINTERNAL="No">
            <TOPPARTS>First TDL Part</TOPPARTS>
        </FORM>
        .
        .
        .
        <FIELD NAME="First TDL Field" ISMODIFY="No"
            ISFIXED="No" ISINITIALIZE="No" ISOPTION="No"
            ISINTERNAL="No">
            <SET>"Welcome to the world of TDL"</SET>
        </FIELD>
    </TDLMESSAGE>
</TDL>
```

TDL request should be enclosed within <TDL> tags. The <TDLMESSAGE> tag is mandatory inside the <TDL> tag. Inside that we can write all the definitions and its attributes with their

values. All definitions and attributes are represented as tags. Consider the following examples which demonstrate the usage of <HEADER> values:

Report specification in TDL

```
<HEADER>
  <VERSION>1</VERSION>
  <TALLYREQUEST>Export</TALLYREQUEST>
  <TYPE>Data</TYPE>
  <ID>Report Name </ID>
</HEADER>
```

In the above header format the value of TallyRequest is Export and the Type is data. Specifying the value of ID is ReportName. This report name should come inside the tag <REPORT> within the <TDL> tag.

For Eg:

```
<TDL>
  <TDLMESSAGE>
    <REPORT NAME="TDL Report" ISMODIFY="No" ISFIXED="No"
      ISINITIALIZE="No" ISOPTION="No" ISINTERNAL="No">
      <FORMS>First TDL Form</FORMS>
    </REPORT>
    .
    .
  </TDLMESSAGE>
</TDL>
```

Collection specification in TDL

```
<HEADER>
  <VERSION>1</VERSION>
  <TALLYREQUEST>Export</TALLYREQUEST>
  <TYPE>Collection</TYPE>
  <ID>Collection Name</ID>
</HEADER>
```

In the above template the value of TallyRequest is Export and the Type is Collection. Specifying the value of ID is CollectionName. This collection name should come inside the tag <COLLECTION> within the <TDL> tag.

```
<TDL>

  <TDLMESSAGE>
    <COLLECTION NAME="Collection of Ledgers"
      ISMODIFY="No" ISFIXED="No" ISINITIALIZE="No"
      ISOPTION="No" ISINTERNAL="No">
      <TYPE>Ledger</TYPE>
    </COLLECTION>
    .
    .
    .
  </TDLMESSAGE>
</TDL>
```

Object specification in TDL

```
<HEADER>
  <VERSION>1</VERSION>
  <TALLYREQUEST>Export</TALLYREQUEST>
  <TYPE>Object</TYPE>
  <ID>Object Name</ID>
</HEADER>
```

In the above template the value of TallyRequest is Export and the Type is Object. Specifying the value of ID is ObjectName. This object name should come inside the tag <OBJECT> within the <TDL> tag.

Consider the following example:

```
<TDL>

  <TDLMESSAGE>
    <OBJECT NAME="Ledger" ISINITIALIZE="Yes">
      <LOCALFORMULA>
        TNetBalance: $$AsPositive: $$AmountSubtract:
          $ClosingBalance: $OpeningBalance
      </LOCALFORMULA>
    </OBJECT>
  </TDLMESSAGE>
</TDL>
```

Function specification in TDL

```
<HEADER>

    <VERSION>1</VERSION>

    <TALLYREQUEST>Export</TALLYREQUEST>

    <TYPE>Function</TYPE>

    <ID>Function Name</ID>

</HEADER>
```

In the above template the value of TallyRequest is Export and the Type is Function. Specifying the value of ID is FunctionName.

```
<DESC>

    <FUNCPARAMLIST>

        <PARAM>@@FirstParameter</PARAM>

        <PARAM TYPE="Number">0.10</PARAM>

    </FUNCPARAMLIST>

    <TDL>

        <TDLMESSAGE>

            <SYSTEM TYPE="Formulae" NAME="FirstParameter" >

                1242849 / 1000

            </SYSTEM>

        </TDLMESSAGE>

    </TDL>

</DESC>
```

In the above example, the function parameter list enclosed with the tag <FUNCPARAMLIST>. The formula which is used inside the Function parameter list is specified inside TDL Tag

<DATA>: Data required for the Request/Response

Data contains the actual data being transferred from one system to another. If the information is retrieved then the data will be obtained inside the <DATA> tag.

```
<DATA>

    <COLLECTION>

        <OBJECT NAME="CDROM Disks 10s - Defective">

            <NAME TYPE="String">CDROM Disks 10s - Defective</NAME>

        </OBJECT>

        <OBJECT NAME="TVS MSP 245 132 Col Printer">

            <NAME TYPE="String">TVS MSP 245 132 Col Printer</NAME>
```

```

        </OBJECT>
        <OBJECT NAME="Assembled PIV">
            <NAME TYPE="String">Assembled PIV</NAME>
        </OBJECT>
    </COLLECTION>
</DATA>

```

In the case of importing data to be sent to Tally, to be specified within the <DATA> tag.

```

<DATA>
    <TALLYMESSAGE>
        <LEDGER Action = "Create" >
            <NAME>ICICI Test</NAME>
            <PARENT>Bank Accounts</PARENT>
            <OPENINGBALANCE>13500</OPENINGBALANCE>
        </LEDGER>
    </TALLYMESSAGE>
</DATA>

```

After having a clear idea about all the important tags and their usage, will concentrate on some examples.

1.7 Case Study – Using the above XML Request/Response Formats

Let us consider different scenarios to understand the Request and Response XML structure.

- Export
 - Request to Export Data and Corresponding response
 - Request to Export different TDL components and the corresponding response
- Import
 - Request to Import Data and the corresponding response
- Execute
 - Request to Execute Action and Corresponding response

1.7.1 Export

Request to Export Data & Corresponding Response

Tags used for sending a request to export data from Tally.ERP 9

<HEADER> contains the following:

- Tag <TALLYREQUEST> must contain value Export
- Tag <TYPE> must contain value Data and
- Tag <ID> should contain the Name of the TDL Report

<BODY> contains the following:

- Tag <DESC> can contain report settings like Company Name, Format, etc. as desired which should be enclosed within <STATICVARIABLES> tag.
- If the Report Name specified in the <ID> tag does not exist within Tally running at the specified port, the TDL defining the Report & other supporting definition needs to be described and enclosed within tag <TDL>.

Request for a detailed Trial Balance in XML Format from Tally

- **Where Report exists in Tally**

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Export</TALLYREQUEST>
    <TYPE>Data</TYPE>
    <ID>Trial Balance</ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        <EXPLODEFLAG>Yes</EXPLODEFLAG>
        <SVEXPORTFORMAT>$$SysName:XML</SVEXPORTFORMAT>
      </STATICVARIABLES>
    </DESC>
  </BODY>
</ENVELOPE>
```

Example 1.1 Request for Trial Balance in XML Format

In the above XML request, <HEADER> describes the expected result.

- The value of the Tag <TALLYREQUEST> is Export which indicates that some information needs to be exported from Tally.
- The value of the Tag <TYPE> is Data which indicates that the data needs to be exported from Tally.
- The value of the Tag <ID> must be a TDL Report Name, if the previous Tag <TYPE> contains Data and Tag <TALLYREQUEST> contains Export. Any Report which needs to be exported from Tally can be specified within this Tag.

- <BODY> Tag contains parameters, if any. Additional settings for the report like format required, company from which data is required, etc. can be passed within <STATICVARIABLES> Tag enclosed within <DESC> Tag. All variables are considered as Tag Names and their value are enclosed within these tags. For e.g., in the above XML, variables SVEXPORTFORMAT and EXPLODEFLAG are considered as Tags and their respective values \$\$SysName:XML and Yes are enclosed within. TDL Internal Function SysName is evaluated at Tally end and the response is being sent accordingly.

XML Response received is as shown:

```
<ENVELOPE>
  <DSPACCNAME>
    <DSPDISPNAME>Capital Account</DSPDISPNAME>
  </DSPACCNAME>
  <DSPACCINFO>
    <DSPCLDRAMT>
      <DSPCLDRAMTA></DSPCLDRAMTA>
    </DSPCLDRAMT>
    <DSPCLCRAMT>
      <DSPCLCRAMTA>100000.00</DSPCLCRAMTA>
    </DSPCLCRAMT>
  </DSPACCINFO>
  </DSPACCNAME>
  <DSPACCINFO>
    <DSPCLDRAMT>
      <DSPCLDRAMTA>-100000.00</DSPCLDRAMTA>
    </DSPCLDRAMT>
    <DSPCLCRAMT>
      <DSPCLCRAMTA></DSPCLCRAMTA>
    </DSPCLCRAMT>
  </DSPACCINFO>
  .
  .
  .
</ENVELOPE>
```

Figure 1.2 Response- Trial Balance in XML Format



Apart from XML, <SVEXPORTFORMAT>Tag can contain the values \$\$SysName:HTML, \$\$SysName:ASCII, \$\$SysName:SDF and BinaryXML.

□ Where Report do not exist in Tally

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Export</TALLYREQUEST>
    <TYPE>Data</TYPE>
    <ID>Simple Trial balance</ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        <EXPLODEFLAG>Yes</EXPLODEFLAG>
        <SVEXPORTFORMAT>$$SysName:XML</SVEXPORTFORMAT>
      </STATICVARIABLES>
      <TDL>
        <TDLMESSAGE>
          <REPORT NAME="Simple Trial balance">
            <FORMS>Simple Trial balance</FORMS>
            <TITLE>"Trial Balance"</TITLE>
          </REPORT>
          <FORM NAME="Simple Trial balance">
            <TOPPARTS>Simple TB Part</TOPPARTS>
            <HEIGHT>100% Page</HEIGHT>
            <WIDTH>100% Page</WIDTH>
          </FORM>
          <PART NAME="Simple TB Part">
            <TOPLINES>Simple TB Title,
              Simple TB Details</TOPLINES>
            <REPEAT>
```



```

Simple TB Details : Simple TB Ledgers
</REPEAT>
<SCROLLED>Vertical</SCROLLED>
<COMMONBORDERS>Yes</COMMONBORDERS>
</PART>
<LINE NAME="Simple TB Title">
  <USE>Simple TB Details</USE>
  <LOCAL>
    Field : Default : Type : String
  </LOCAL>
  <LOCAL>
    Field : Default : Align : Centre
  </LOCAL>
</LOCAL>
Field : Simple TB Name Field : Set as: "Particulars"
</LOCAL>
<LOCAL>
  Field : Simple TB Amount Field: Set as: "Amount"
</LOCAL>
<BORDER>Flush Totals</BORDER>
</LINE>
<LINE NAME="Simple TB Details">
  <LEFTFIELDS>Simple TB Name Field</LEFTFIELDS>
  <RIGHTFIELDS>Simple TB Amount Field</RIGHTFIELDS>
</LINE>
  <FIELD NAME="Simple TB Name Field">
    <USE>Name Field</USE>
    <SET>$Name</SET>
  </FIELD>
  <FIELD NAME="Simple TB Amount Field">
    <USE>Amount Field</USE>
    <SET>$ClosingBalance</SET>
    <BORDER>Thin Left</BORDER>
  </FIELD>
<COLLECTION NAME="Simple TB Ledgers">
  <TYPE>Ledger</TYPE>

```

```

        <FILTERS>NoProfitsimple</FILTERS>
    </COLLECTION>
    <SYSTEM TYPE="Formulae" NAME="NoProfitSimple">
        NOT $$IsLedgerProfit
    </SYSTEM>
</TDLMESSAGE>
</TDL>
</DESC>
</BODY>
</ENVELOPE>

```

The above XML Request is similar to the previous Trial Balance Report request. The difference is the Report Name contained within the <ID> Tag is not defined in Tally.

- In the <BODY> Tag within <DESC> Tag, an additional tag <TDL> must be specified with the TDL describing the Report and its components enclosed within Tag <TDLMESSAGE>.

XML Response received is as shown:

```

<ENVELOPE>
    <SIMPLETBNAMEFIELD>Bank of India</SIMPLETBNAMEFIELD>
    <SIMPLETBAMOUNTFIELD>351265.00</SIMPLETBAMOUNTFIELD>
    <SIMPLETBNAMEFIELD>Cash</SIMPLETBNAMEFIELD>
    <SIMPLETBAMOUNTFIELD>-147600.00</SIMPLETBAMOUNTFIELD>
    <SIMPLETBNAMEFIELD>Conveyance</SIMPLETBNAMEFIELD>
    <SIMPLETBAMOUNTFIELD>-157665.00</SIMPLETBAMOUNTFIELD>
    ,
    ,
</ENVELOPE>

```

Request to Export different TDL components & Corresponding Response

The different TDL components used for exporting are:

- Object
- Collection
- Function

Request To Export Object & Corresponding Response

For sending a request to export an Object Info from Tally,

<HEADER> contains the following

- ❑ Tag <TALLYREQUEST> must contain value Export
- ❑ Tag <TYPE> must contain value Object
- ❑ Tag <SUBTYPE> must contain the Type of Object and
- ❑ Tag <ID> should contain the Object Identifier

<BODY> contains the following within <DESC> Tag

- ❑ Settings like Company Name, Format to be exported, etc. as desired enclosed within <STATICVARIABLES> Tag.
- ❑ Methods that need to be retrieved within the <FETCHLIST> Tag under each <FETCH> Tag.
- ❑ External Methods, if any, must be specified within <LOCAL FORMULA> Tag enclosed within <OBJECT> Tag.
- ❑ Above Local Formula, if dependent on any Local or Global Formula needs to be specified

Request for an Object info in XML Format from Tally.ERP 9

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>OBJECT</TYPE>
    <SUBTYPE>Ledger</SUBTYPE>
    <ID TYPE="Name">ABC India Pvt. Ltd. </ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        <SVEXPORTFORMAT>$$SysName:XML</SVEXPORTFORMAT>
      </STATICVARIABLES>
      <FETCHLIST>
        <FETCH>Name</FETCH>
        <FETCH>TNetBalance</FETCH>
        <FETCH>LedgerPhone</FETCH>
      </FETCHLIST>
    <TDL>
```

```

        <TDLMESSAGE>
            <OBJECT NAME="Ledger" ISINITIALIZE="Yes">
                <LOCALFORMULA>
                    TNetBalance: $$AsPositive:
                    $$AmountSubtract: $ClosingBalance:
                    $OpeningBalance
                </LOCALFORMULA>
            </OBJECT>
        </TDLMESSAGE>
    </TDL>
</DESC>
</BODY>
</ENVELOPE>

```

Figure 1.3 Request- Methods of a Ledger Object in XML Format

In the above XML request, <HEADER> describes the expected result.

- ❑ The value of the Tag <TALLYREQUEST> is Export which indicates that some information needs to be exported from Tally.
- ❑ The value of the Tag <TYPE> is Object which indicates that information pertaining to some Object needs to be exported from Tally.
- ❑ The value of the Tag <SUBTYPE> is Ledger which indicates that Ledger Object info needs to be exported from Tally.
- ❑ The value of the Tag <ID> must contain the Ledger Identifier which is the name of the ledger
- ❑ The <BODY> Tag contains description within <DESC> Tag which requires all info pertaining to the Object required

XML Response received is as shown:

```

<ENVELOPE>
    <HEADER>
        <VERSION>1</VERSION>
        <STATUS>1</STATUS>
    </HEADER>
    <BODY>
        <DESC>
        </DESC>
        <DATA>

```

```

<TALLYMESSAGE>
<LEDGER NAME="ABC India Pvt. Ltd." RESERVEDNAME="">
<NAME.LIST TYPE="String">
    <NAME TYPE="String">ABC India Pvt. Ltd.</NAME>
    <NAME/>
</NAME.LIST>
<RESERVEDNAME TYPE="String"></RESERVEDNAME>
<LEDGERPHONE TYPE="String">9940421583</LEDGERPHONE>
<TNETBALANCE TYPE="Amount">-13240.00</TNETBALANCE>
</LEDGER>
</TALLYMESSAGE>
</DATA>
</BODY>
</ENVELOPE>

```

Figure 1.4 Response- Methods of a Ledger Object in XML Format

The above response has been received based on the XML request specified. The required detail of the Ledger "ABC India Pvt Ltd" i.e., Phone No, Contact, Opening, Closing has been sent from Tally.

Request To Export Collection & Corresponding Response

For sending a request to export Collection data from Tally,

<HEADER> contains the following

- ❑ Tag <TALLYREQUEST> must contain value Export
- ❑ Tag <TYPE> must contain value Collection
- ❑ Tag <ID> should contain the Collection Name which is being described within the <DESC> Tag

<BODY> contains the following within <DESC> Tag

- ❑ Tag <DESC> can contain settings like Company Name, Format, etc. as desired which should be enclosed within <STATICVARIABLES> tag.
- ❑ Collection declared within the <HEADER> tag <TYPE> must be defined within <TDLMESSAGE> tag under <TDL> tag.
- ❑ All the TDL Collection Attributes must be specified as tags and their respective values within the relevant tag.

Where Collection exists in Tally

Request for Collection Data in XML Format from Tally.ERP 9

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>COLLECTION</TYPE>
    <ID>Remote Ledger Coll</ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        <SVEXPORTFORMAT>$$SysName:XML</SVEXPORTFORMAT>
      </STATICVARIABLES>
      <TDL>
        <TDLMESSAGE>
          <COLLECTION NAME="Remote Ledger Coll"
            ISINITIALIZE="Yes">
              <TYPE>Ledger</TYPE>
              <NATIVEMETHOD>Name</NATIVEMETHOD>
              <NATIVEMETHOD>OpeningBalance
            </NATIVEMETHOD>
          </COLLECTION>
        </TDLMESSAGE>
      </TDL>
    </DESC>
  </BODY>
</ENVELOPE>
```

Figure 1.5 Request- Collection data in XML Format

In the above XML request, <HEADER> describes the expected result.

- ❑ The value of the Tag <TALLYREQUEST> is Export which indicates that some information needs to be exported from Tally.
- ❑ The value of the Tag <TYPE> is Collection which indicates that information pertaining to Collection needs to be exported from Tally.
- ❑ The value of the Tag <ID> must contain the Collection Name which is defined available Tally

Response in XML

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>1</STATUS>
  </HEADER>
  <BODY>
    <DESC>
    </DESC>
    <DATA>
      <STOCKITEM NAME="Item" RESERVEDNAME="">
        <LANGUAGENAME.LIST>
          <NAME.LIST TYPE="String">
            <NAME>Item</NAME>
          </NAME.LIST>
          <LANGUAGEID TYPE="Number"> 1033</LANGUAGEID>
        </LANGUAGENAME.LIST>
      </STOCKITEM>
    </COLLECTION>
  </DATA>
</BODY>
</ENVELOPE>
```

Figure 1.6 Response- Collection data in XML Format

□ Where Collection does not exist in Tally

In such case, Collection Tag must be defined along with their attributes as sub tags inside the Body Tag enclosed within TDL Message Tag.

Request for Collection Data in XML Format from Tally.ERP 9

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>COLLECTION</TYPE>
    <ID>Remote Ledger Coll</ID>
  </HEADER>
  <BODY>
    <DESC>
```

```

<STATICVARIABLES>
    <SVEXPORTFORMAT>$$SysName:XML</SVEXPORTFORMAT>
</STATICVARIABLES>
<TDL>
<TDLMESSAGE>
    <COLLECTION NAME="Remote Ledger Coll"
        ISINITIALIZE="Yes">
        <TYPE>Ledger</TYPE>
        <NATIVEMETHOD>Name</NATIVEMETHOD>
        <NATIVEMETHOD>OpeningBalance</NATIVEMETHOD>
    </COLLECTION>
</TDLMESSAGE>
</TDL>
</DESC>
</BODY>
</ENVELOPE>

```

Figure 1.7 Request- Collection data in XML Format

In the above XML request, <HEADER> describes the expected result.

- The value of the Tag <TALLYREQUEST> is Export which indicates that some information needs to be exported from Tally.
- The value of the Tag <TYPE> is Collection which indicates that information pertaining to Collection needs to be exported from Tally.
- The value of the Tag <ID> must contain the Collection Name which is defined below in <TDLMESSAGE> Tag within <DESC> Tag under the Tag <BODY>.

Response XML Fragment for the above :

```

<ENVELOPE>
    <HEADER>
        <VERSION>1</VERSION>
        <STATUS>1</STATUS>
    </HEADER>
    <BODY>
        <DESC>
        </DESC>
        <DATA>
            <COLLECTION>

```



```

<LEDGER NAME="ABC India Pvt. Ltd." RESERVEDNAME="">
  <OPENINGBALANCE TYPE="Amount">
    5000.00
  </OPENINGBALANCE>
  <LANGUAGEName.LIST>
    <NAME.LIST TYPE="String">
      <NAME>ABC India Pvt. Ltd.</NAME>
    </NAME.LIST>
    <LANGUAGEID TYPE="Number"> 1033</LANGUAGEID>
  </LANGUAGEName.LIST>
</LEDGER>
<LEDGER NAME="XYZ Loan A/c" RESERVEDNAME="">
  <OPENINGBALANCE TYPE="Amount">
    0.00
  </OPENINGBALANCE>
  <LANGUAGEName.LIST>
    <NAME.LIST TYPE="String">
      <NAME>XYZ Loan A/c</NAME>
    </NAME.LIST>
    <LANGUAGEID TYPE="Number"> 1033</LANGUAGEID>
  </LANGUAGEName.LIST>
</LEDGER>
<LEDGER NAME="Accum. Dep. on Airconditioner" RESERVEDNAME="">
  <OPENINGBALANCE TYPE="Amount">0.00</OPENINGBALANCE>
  <LANGUAGEName.LIST>
    <NAME.LIST TYPE="String">
      <NAME>Accum. Dep. on Airconditioner</NAME>
    </NAME.LIST>
    <LANGUAGEID TYPE="Number"> 1033</LANGUAGEID>
  </LANGUAGEName.LIST>
</LEDGER>
</COLLECTION>
</DATA>
</BODY>
</ENVELOPE>

```

Figure 1.8 Response - Collection data in XML Format

The above response has been received based on the XML request specified. All the Ledgers with the required Methods i.e., Name and Opening Balance are sent from Tally.

Request To Export Function & Corresponding Response

For sending a request to evaluate the result of Function from Tally.ERP 9

<HEADER> contains the following

- Tag <TALLYREQUEST> must contain value Export
- Tag <TYPE> must contain value Function
- Tag <ID> should contain the Function which is being described within the <DESC> Tag within the Tag <BODY>

<BODY> contains the following:

- Tag <DESC> can contain settings like Company Name, Format, etc. as desired which should be enclosed within <STATICVARIABLES> Tag.

Request for evaluating function in Tally without parameter

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>FUNCTION</TYPE>
    <ID>$$NumStockItems</ID>
  </HEADER>
  <BODY>
    <DESC>
  </DESC>
  </BODY>
</ENVELOPE>
```

Figure 1.9 Request- Function without Parameter evaluation

In the above XML request,

- <HEADER> describes the expected result.
- The value of the Tag <TALLYREQUEST> is Export which indicates that some information needs to be exported from Tally.
- The value of the Tag <TYPE> is Function which indicates that some Function needs to be evaluated within Tally and some value is returned as a response in XML.
- The value of the Tag <ID> must contain the Function Name prefixed with \$\$ since Function in TDL is activated by \$\$.

Response XML Fragment for the above would be

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>1</STATUS>
  </HEADER>
  <BODY>
    <DESC>
    </DESC>
    <DATA>
      <RESULT TYPE="Number">33</RESULT>
    </DATA>
  </BODY>
</ENVELOPE>
```

Figure 1.10 Response- Function without Parameter evaluation

Request for evaluating function with Parameters in Tally

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>EXPORT</TALLYREQUEST>
    <TYPE>FUNCTION</TYPE>
    <ID>$$Round</ID>
  </HEADER>
  <BODY>
    <DESC>
      <FUNCPARAMLIST>
        <PARAM>@@FirstParameter</PARAM>
        <PARAM TYPE="Number">0.10</PARAM>
      </FUNCPARAMLIST>
      <TDL>
        <TDLMESSAGE>
          <SYSTEM TYPE="Formulae"
            NAME="FirstParameter" >
              1242849 / 1000
          </SYSTEM>
```

```

        </TDLMESSAGE>
    </TDL>
    </DESC>
</BODY>
</ENVELOPE>

```

Figure 1.11 Request- Function evaluation with Parameters

The above XML request is similar to the previous request except for this Function evaluation request needs Parameters to be specified.

- All the Parameters must be specified within the Tag <FUNCPARAMLIST> in the <DESC> Tag under <BODY> Tag.
- Each parameter must be enclosed within <PARAM> Tag.
- Parameters must follow exactly in the order required by the Function which is specified in the <ID> Tag.
- Dependency, if any i.e., Global/System Formula must be defined with <TDLMESSAGE> Tag under Tag <TDL>.

Response XML Fragment for the above would be:

```

<ENVELOPE>
    <HEADER>
        <VERSION>1</VERSION>
        <STATUS>1</STATUS>
    </HEADER>
    <BODY>
        <DESC>
        </DESC>
        <DATA>
            <RESULT TYPE="Number">1242.80</RESULT>
        </DATA>
    </BODY>
</ENVELOPE>

```

Figure 1.12 Response- Function evaluation with Parameters

The above response has been generated from Tally based on the request specified. <DATA> Tag contains the <RESULT> Tag which holds the result after function evaluation.

1.7.2 Import

Request to Import Data and Corresponding Response

Tally can import data objects either in the form of a Master or Voucher.

<HEADER> contains the following

- Tag <TALLYREQUEST> must contain value Import
- Tag <TYPE> must contain value Data and
- Tag <ID> should contain the Import TDL Report i.e., either All Masters or Vouchers

<BODY> contains the following

- Tag <DESC> can contain report settings like Company Name, behavior of Import in case of duplicates found; as desired which should be enclosed within <STATICVARIABLES> Tag.
- Tag <DATA> must contain the XML Data Fragment within Tag <TALLYMESSAGE> that needs to be imported

Request for importing Master data in Tally

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Import</TALLYREQUEST>
    <TYPE>Data</TYPE>
    <ID>All Masters</ID>
  </HEADER>
  <BODY>
    <DESC>
      <STATICVARIABLES>
        <IMPORTDUPS>@@DUPCOMBINE</IMPORTDUPS>
      </STATICVARIABLES>
    </DESC>
    <DATA>
      <TALLYMESSAGE>
        <LEDGER NAME="ICICI" Action = "Create">
          <NAME>ICICI</NAME>
          <PARENT>Bank Accounts</PARENT>
          <OPENINGBALANCE>-12500</OPENINGBALANCE>
        </LEDGER>
        <GROUP NAME=" Bangalore Debtors" Action = "Create">
          <NAME>Bangalore Debtors</NAME>
```

```
<PARENT>Sundry Debtors</PARENT>
</GROUP>
<LEDGER NAME="RK Builders Pvt Ltd" Action = "Create">
  <NAME>RK Builders Pvt Ltd</NAME>
  <PARENT>Bangalore Debtors</PARENT>
  <OPENINGBALANCE>-1000</OPENINGBALANCE>
</LEDGER>
</TALLYMESSAGE>
</DATA>
</BODY>
</ENVELOPE>
```

Figure 1.13 Request- Import Master in Tally

In the above XML Request, Create action is used. Any of the following system formulae can be used to choose the required behaviour in case the system encounters a ledger with the same name. The behavior is for the treatment of the Opening Balance which is being imported.

DupModify specifies that the current Opening Balance should be modified with the new one that is being imported.

DupIgnoreCombine specifies that the ledger if exists need to be ignored.

DupCombine specifies the system to combine both the Opening Balances. Ideally, this option is used when Data pertaining to Group Companies are merged together into a single company.

On processing the above request for importing ledgers, the requested ledgers are created in Tally and the following response is received:

```
<RESPONSE>
  <CREATED>2</CREATED>
  <ALTERED>0</ALTERED>
  <LASTVCHID>0</LASTVCHID>
  <LASTMID>0</LASTMID>
  <COMBINED>0</COMBINED>
  <IGNORED>0</IGNORED>
  <ERRORS>0</ERRORS>
</RESPONSE>
```

Figure 1.14 Response - After Ledger Master Import in Tally

The above XML Response is a log of masters created, altered, combined, ignored or not imported due to some errors. It also contains information pertaining to the last Master ID imported.

For Alteration and Deletion of Masters, the Object action needs to be Alter or Delete respectively.

For instance, in the above example,

```
<LEDGER NAME="ICICI" Action = "Alter">
    <NAME>HDFC</NAME>
```

Name of an existing ledger ICICI will get altered to HDFC.

In case of Deletion, following line suffices

```
<LEDGER NAME="ICICI" Action = "Delete">
```

Request for importing Voucher in Tally (Voucher Creation)

```
<ENVELOPE>
    <HEADER>
        <VERSION>1</VERSION>
        <TALLYREQUEST>Import</TALLYREQUEST>
        <TYPE>Data</TYPE>
        <ID>Vouchers</ID>
    </HEADER>
    <BODY>
        <DESC>
        </DESC>
        <DATA>
            <TALLYMESSAGE>
                <VOUCHER>
                    <DATE>20080402</DATE>
                    <NARRATION>Ch. No. Tested</NARRATION>
                    <VOUCHERTYPENAME>Payment</VOUCHERTYPENAME>
                    <VOUCHERNUMBER>1</VOUCHERNUMBER>
                    <ALLEDGERENTRIES.LIST>
                        <LEDGERNAME>Conveyance</LEDGERNAME>
                        <ISDEEMEDPOSITIVE>Yes</ISDEEMEDPOSITIVE>
                        <AMOUNT>-12000.00</AMOUNT>
                    </ALLEDGERENTRIES.LIST>
                    <ALLEDGERENTRIES.LIST>
                        <LEDGERNAME>Bank of India</LEDGERNAME>
                        <ISDEEMEDPOSITIVE>No</ISDEEMEDPOSITIVE>
                        <AMOUNT>12000.00</AMOUNT>
                    </ALLEDGERENTRIES.LIST>
                </VOUCHER>
            </TALLYMESSAGE>
        </DATA>
    </BODY>
</ENVELOPE>
```

```

</VOUCHER>
<VOUCHER>
  <DATE>20080402</DATE>
  <NARRATION>Ch. No. : Tested</NARRATION>
  <VOUCHERTYPENAME>Payment</VOUCHERTYPENAME>
  <VOUCHERNUMBER>2</VOUCHERNUMBER>
  <ALLEDGERENTRIES.LIST>
    <LEDGERNAME>Conveyance</LEDGERNAME>
    <ISDEEMEDPOSITIVE>Yes</ISDEEMEDPOSITIVE>
    <AMOUNT>-5000.00</AMOUNT>
  </ALLEDGERENTRIES.LIST>
  <ALLEDGERENTRIES.LIST>
    <LEDGERNAME>Bank of India</LEDGERNAME>
    <ISDEEMEDPOSITIVE>No</ISDEEMEDPOSITIVE>
    <AMOUNT>5000.00</AMOUNT>
  </ALLEDGERENTRIES.LIST>
</VOUCHER>
</TALLYMESSAGE>
</DATA>
</BODY>
</ENVELOPE>

```

Figure 1.15 Request- Import Voucher in Tally - Creation

On processing the above request for importing vouchers, the requested vouchers are created in Tally and the following response is received:

```

<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>1</STATUS>
  </HEADER>
  <BODY>
    <DATA>
      <IMPORTRESULT>
        <CREATED>2</CREATED>
        <ALTERED>0</ALTERED>
        <LASTVCHID>119</LASTVCHID>
        <LASTMID>0</LASTMID>
      </IMPORTRESULT>
    </DATA>
  </BODY>
</ENVELOPE>

```



```

        <COMBINED>0</COMBINED>
        <IGNORED>0</IGNORED>
        <ERRORS>0</ERRORS>
    </IMPORTRESULT>
</DATA>
</BODY>
</ENVELOPE>

```

Figure 1.16 Response- After Voucher Import in Tally - Creation

The above XML Response is a log of vouchers created, altered, combined, ignored or not imported due to some errors. It also contains information pertaining to last Voucher ID imported.

Request for importing Voucher in Tally (Voucher Alteration)

In case of Voucher Alteration, Cancellation or Deletion, vital information required is the voucher identifier. Identification of Voucher can be direct Methods within Voucher Object. For example, Master ID, Voucher Number, Reference, Narration, etc. Specifying Voucher Date is mandatory.

```

<ENVELOPE>
    <HEADER>
        <VERSION>1</VERSION>
        <TALLYREQUEST>Import</TALLYREQUEST>
        <TYPE>Data</TYPE>
        <ID>Vouchers</ID>
    </HEADER>
    <BODY>
        <DESC>
        </DESC>
        <DATA>
            <TALLYMESSAGE>
                <VOUCHER DATE="02-Apr-2008" TAGNAME = "Voucher Number"
                    TAGVALUE="3" Action="Alter" VCHTYPE = "Sales">
                    <DATE>20080402</DATE>
                    <NARRATION>Being Goods sold</NARRATION>
                </VOUCHER>
            </TALLYMESSAGE>
        </DATA>
    </BODY>
</ENVELOPE>

```

Figure 1.17 Request- Import Voucher in Tally - Alteration

Request for importing Voucher in Tally (Voucher Cancellation)

Voucher cancellation is similar to above Voucher Alteration. For Voucher Cancellation, Action must be set to "Cancel"

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Import</TALLYREQUEST>
    <TYPE>Data</TYPE>
    <ID>Vouchers</ID>
  </HEADER>
  <BODY>
    <DESC>
    </DESC>
    <DATA>
      <TALLYMESSAGE>
        <VOUCHER DATE="02-Apr-2008" TAGNAME = "Voucher Number"
          TAGVALUE="3" VCHTYPE = "Sales" ACTION="Cancel">
          <NARRATION>
            Being cancelled due to XYZ Reasons
          </NARRATION>
        </VOUCHER>
      </TALLYMESSAGE>
    </DATA>
  </BODY>
</ENVELOPE>
```

Figure 1.18 Request- Import Voucher in Tally - Cancellation

Request for importing Voucher in Tally (Voucher Deletion)

Voucher Deletion is similar to above Voucher Alteration or Cancellation. For Voucher Deletion, Action must be set to "Delete".

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Import</TALLYREQUEST>
    <TYPE>Data</TYPE>
    <ID>Vouchers</ID>
  </HEADER>
```

```

<BODY>
  <DESC>
</DESC>
  <DATA>
    <TALLYMESSAGE>
      <VOUCHER DATE="02-Apr-2008" TAGNAME = "Voucher Number"
        TAGVALUE="3" VCHTYPE = "Sales" ACTION="Delete">
    </VOUCHER>
  </TALLYMESSAGE>
</DATA>
</BODY>
</ENVELOPE>

```

Figure 1.19 Request- Import Voucher in Tally - Deletion

1.7.3 Execute

Request to Execute Action & Corresponding Response

Tags used for sending a request to Execute an action from Tally.ERP 9

<HEADER> contains the following:

- ❑ Tag <TALLYREQUEST> must contain value Execute
- ❑ Tag <TYPE> must contain value TDLAction and
- ❑ Tag <ID> should contain the Name of the TDL Action

As of now only Sync action is introduced. For sync no parameters are required.

Request for Executing Synchronization in Tally

```

<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <TALLYREQUEST>Execute</TALLYREQUEST>
    <TYPE>TDLAction</TYPE>
    <ID>Sync</ID>
  </HEADER>
</ENVELOPE>

```

Figure 1.20 Request for synchronization in XML format

In the above XML request, <HEADER> describes the expected result.

- ❑ The value of the Tag <TALLYREQUEST> is Execute which indicates that some action needs to be executed in Tally.
- ❑ The value of the Tag <TYPE> is TDLAction which indicates that some TDLAction has to be executed in Tally.
- ❑ The value of the Tag <ID> must be a TDL Action Name. Any action which needs to be executed in Tally can be specified within this Tag.

Response XML Fragment for the above would be:

```
<ENVELOPE>
  <HEADER>
    <VERSION>1</VERSION>
    <STATUS>1</STATUS>
  </HEADER>
</ENVELOPE>
```



Ensure the following while executing the sync action:

- ❑ All synchronization setup has to be done at server end as well as client end
- ❑ Set the Option **Enable ODBC server** in Advanced configuration to **Yes**
- ❑ Pass the **Request** from the client end only

Lesson2: Integration Using XML Interface

On the completion of this chapter you will be able to

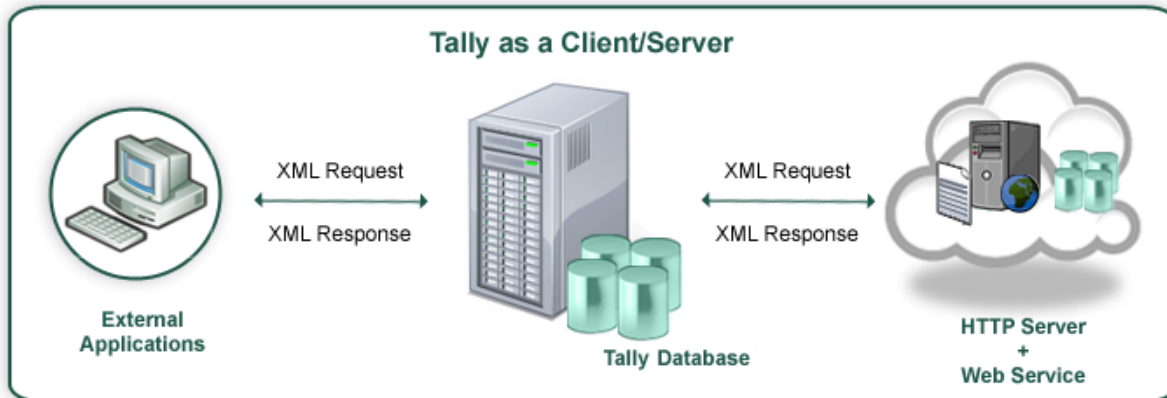
- Understand the functionality of Tally as a Server using External applications
- Understand the functionality of Tally as a Client using Web Services

Introduction

Tally.ERP 9 has supported integration with web scripting languages such as ASP/Perl/PHP and other languages like VB or any environment capable of supporting XML and HTTP. Integration with these products is possible as XML import and export capability is built into Tally.ERP 9.

In fact, Tally.ERP 9 delivers most of the functionalities of Web Services provided by Microsoft's .NET framework. All Tally.ERP 9 data is accessible to any number of potentially disparate systems through the use of Internet standards such as XML and HTTP. In other words, Tally.ERP 9 can communicate with any environment capable of sending and receiving XML over HTTP.

This chapter explains how Tally.ERP 9 will act as a server/client while it is connecting to external applications. The following figure shows the XML Messaging Format through external application/ Web services, acting Tally.ERP 9 as a Server/Client.



2.1 Tally.ERP 9 as a Server – Using External application as Front End

Data can be accessed from Tally.ERP 9 once the connection is established between Tally.ERP 9 and other external applications. Here we are using Data can be posted from Visual Basic to Tally.ERP 9 through XML Interface. The existing Tally.ERP 9 data can be altered and deleted from Visual Basic. Let us discuss some scenarios for using external application as front end.

2.1.1 Case Study I – Importing Masters from Excel to Tally.ERP 9

A Company "Global Enterprises" was using external software as on 31st March 2008. As on 1st April 2008, they have procured Tally.ERP 9. There is a requirement for all their ledgers and inventory masters to be transferred to Tally without entering them. The external software has an option to export its master data to Excel. Using the same, all ledger and inventory masters have been exported in Excel Sheets. The data files thus exported are displayed below.

ID	Customer Name	Group	Building	Road	City	State	Phone	Fax	Email
C1	Ram & Co.	Sundry Debtors	Raheja Arcade	Koramangala	Bangalore	Karnataka	24567881	24567882	ram@yahoo.com
C2	Syscon Computers Ltd.	Sundry Debtors	Swati Mansion	M.G.Road	Managalore	Karnataka	27894152	27894153	syscon@gmail.com
C3	Airtel Communications	Sundry Creditors	Shivam Complex	Rajaji Path	Mumbai	Maharashtra	26541235	26541236	airtel@airtel.in
C4	Shakti Traders	Sundry Debtors	D'costa Mansion	Federick Road	Panaji	Goa	28492788	28492789	shakti@rediff.com
C5	Emco Transformers Pvt. Ltd.	Sundry Creditors	B'Ganza Towers	Roderick Road	Panaji	Goa	25964125	25964126	emco@emco.com
C6	EMI Transmission	Sundry Debtors	Madhavi Scty	Tilak Nagar	Mumbai	Maharashtra	23451456	23451457	emi@emi.com
C7	Vedha Automation	Sundry Debtors	Hemjibhai Scty	Shanivar Peth	Pune	Maharashtra	23474455	23474456	vedha@vedha.com

Figure 1.1 Ledger Master

ItemID	Item	Group	Unit	Qty	Rate
SI01	Monitor	Hardware	Nos.	1200	9000
SI02	Mouse	Hardware	Nos.	2400	400
SI03	HardDisk 80 GB	Hardware	Nos.	500	2500
SI04	Keyboard	Hardware	Nos.	1000	900
SI05	Laptop HP	Laptops	Nos.	800	34000
SI06	Laptop HCL	Laptops	Nos.	200	30000

Figure 1.2 Inventory Master

A Company Global Enterprises has been created in Tally.ERP 9 and only default masters exist as shown in the screen below:

Global Enterprises For 1-Apr-2008		Global Enterprises For 1-Apr-2008	
Types of Vouchers		Types of Accounts	
Attendance	0	Groups	28
Contra	0	Ledgers	2
Credit Note	0	Stock Groups	0
Debit Note	0	Stock Items	0
Delivery Note	0	Voucher Types	18
Journal	0	Units	0
Memorandum	0	Currencies	1
Payment	0		
Payroll	0		
Physical Stock	0		
Purchase	0		
Purchase Order	0		
Receipt	0		
Receipt Note	0		
Rejections In	0		
Rejections Out	0		
Reversing Journal	0		
Sales	0		
Sales Order	0		
Stock Journal	0		
Total	0		

Figure 1.3 Statistics prior to Import

The data in Excel needs to be converted to Tally understandable XML format and sent to the port in which Tally.ERP 9 is running.

To achieve this, an interface is built in VB to import all the masters from Excel and generate a Tally compatible XML which is subsequently posted in Tally.ERP 9. The interface application created for the same has been displayed below.



Figure 1.4 Data Transfer to Tally

On selecting Type "Stock Items" or "Ledgers" and clicking on Export button, it transfers all the relevant masters to Tally.ERP 9. On the Export to Tally Button, required XML as discussed earlier is constructed and posted to Tally.ERP 9 which is up at a predefined port.

The XML is generated through the following VB Code Snippet:-

```
Private Function LedgerMasterText(ByVal intI As Integer) As String
    Dim strTemp As String
    Dim strTxt As String
    sbExport.SimpleText = intI & ": " & Trim$(xlWS.Cells(intI, 2))
    strTxt = vbNullString
    strTxt = _
    "<ENVELOPE>" & vbCrLf & _
    "    <HEADER>" & vbCrLf & _
    "        <VERSION>1</VERSION>" & vbCrLf & _
    "        <TALLYREQUEST>Import </TALLYREQUEST>" & vbCrLf & _
    "        <TYPE>Data</TYPE>" & vbCrLf & _
    "        <ID>All Masters</ID>" & vbCrLf & _
    "    </HEADER>" & vbCrLf & _
    "    <BODY>" & vbCrLf & _
    "        <DESC>" & _
    "            <STATICVARIABLES>" & _
    "                <SVCURRENTCOMPANY>" & _
    "                    ##SVCurrentCompany" & _
```



```

        "</SVCURRENTCOMPANY>" & _
        "</STATICVARIABLES>" & _
        "</DESC>" & _
        "<DATA>" & vbCrLf & _
        "<TALLYMESSAGE>" & vbCrLf & _
        "<LEDGER>" & vbCrLf & _
        "<NAME.LIST>" & vbCrLf & _
        "<NAME>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 2))) &
        "</NAME>" & vbCrLf
    If Trim$(xlWS.Cells(intI, 1)) <> vbNullString Then
        strTxt = strTxt & "<NAME>" &
        ReplaceXmlText(Trim$(xlWS.Cells(intI, 1))) & "</NAME>" & vbCrLf
    End If
    strTxt = strTxt & _
        "</NAME.LIST>" & vbCrLf & _
        "<PARENT>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 3))) &
        "</PARENT>" & vbCrLf
    '-----Optional-----
    If Trim$(xlWS.Cells(intI, 4)) <> vbNullString Then
        strTxt = strTxt & "<ADDRESS.LIST>" & vbCrLf
        strTxt = strTxt &
            "<ADDRESS>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 4))) &
            "</ADDRESS>" & vbCrLf & 'Address 1
    If Trim$(xlWS.Cells(intI, 5)) <> vbNullString Then
        strTxt = strTxt & _
            "<ADDRESS>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 5))) &
            "</ADDRESS>" & vbCrLf & 'Address 2
    If Trim$(xlWS.Cells(intI, 6)) <> vbNullString Then
        strTxt = strTxt & _
            "<ADDRESS>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 6))) &
            "</ADDRESS>" & vbCrLf & 'Address 3
        strTxt = strTxt & "</ADDRESS.LIST>" & vbCrLf
    End If

    If Trim$(xlWS.Cells(intI, 7)) <> vbNullString Then
        strTxt = strTxt & _
            "<STATENAME>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 7))) &

```

```

        "</STATENAME>"
    End If
    If Trim$(xlWS.Cells(intI, 8)) <> vbNullString Then
        strTxt = strTxt &
            "<LEDGERPHONE>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 8))) & _
            "</LEDGERPHONE>"
    End If
    If Trim$(xlWS.Cells(intI, 9)) <> vbNullString Then
        strTxt = strTxt &
            "<LEDGERFAX>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 9))) & _
            "</LEDGERFAX>"
    End If
    If Trim$(xlWS.Cells(intI, 10)) <> vbNullString Then
        strTxt = strTxt &
            "<EMAIL>" & ReplaceXmlText(Trim$(xlWS.Cells(intI, 10))) & _
            "</EMAIL>"
    End If
    strTemp = ReplaceXmlText(Trim$(xlWS.Cells(intI, 2)))
    strTxt = strTxt &
        "<ADDITIONALNAME>" & Trim$(strTemp) & _
        "</ADDITIONALNAME>" & _ vbCrLf
    strTxt = strTxt & _
        "</LEDGER>" & vbCrLf & _
        "</TALLYMESSAGE>" & vbCrLf & _
        "</DATA>" & vbCrLf & _
        "</BODY>" & vbCrLf & _
        "</ENVELOPE>" & vbCrLf
    LedgerMasterText = strTxt
End Function

```

The following VB Code Snippet sends the above generated XML Data to Tally.ERP 9 which is running at a predefined port.

```

objXml.open "POST", "http://localhost:9000", False
objXml.send XMLToPost

```

On importing both Ledgers and Inventory Masters, statistics is as shown below:

Global Enterprises For 1-Apr-2008		Global Enterprises For 1-Apr-2008	
Types of Vouchers		Types of Accounts	
Attendance	0	Groups	28
Contra	0	Ledgers	7
Credit Note	0	Stock Groups	2
Debit Note	0	Stock Items	6
Delivery Note	0	Voucher Types	18
Journal	0	Units	1
Memorandum	0	Currencies	1
Payment	0	Tariff Classifications	0
Payroll	0		
Physical Stock	0		
Purchase	0		
Purchase Order	0		
Receipt	0		
Receipt Note	0		
Rejections In	0		
Rejections Out	0		
Reversing Journal	0		
Sales	0		
Sales Order	0		
Stock Journal	0		
Total	0		

Figure 1.5 Statistics post Import

In the above figure, we can observe that 5 Ledgers and 6 Stock Items have been imported from Excel. If there are some errors while Importing, Tally.imp can be referred for Import Log.

2.1.2 Case Study II - Creation and Alteration of Vouchers through VB

A Company Global Enterprises needs to design an interface for entering their receipt vouchers and altering the same, if required. At the end of Voucher Entry, the same needs to be posted to Tally.ERP 9.

The following interface has been designed for Receipt Voucher Entry.

Receipts Creation/Alteration

Voucher Entry for Receipts

CREATE

EXIT

ALTER

On hitting Option Create, it enters a new form designed for Receipt Creation as shown below.

Create

Creation of Receipts

Date: 16/02/2009

Account: Cash

Amount:

Particulars: Emco Transformers Pvt.
Profit & Loss A/c
Ram & Co.
Shakti Traders
Syscon Computers Ltd.
Vedha Automation

Narration:

EXIT

The Masters created in Tally.ERP 9 are collected and being displayed for selection by the user. Collections pertaining to Cash or Bank Ledgers and Party Ledgers have been written in a TDL File which must be associated before executing this VB code.

The screenshot shows a Windows-style window titled "Create" with a subtitle "Creation of Receipts". Inside the window, there are several input fields and two buttons. The "Date" field contains "16/02/2009". The "Account" field is a dropdown menu showing "Cash". The "Amount" field contains "12000". The "Particulars" field is a dropdown menu showing "Emco Transformers f". The "Narration" field contains "Being Cash received". At the bottom of the window, there are two buttons labeled "POST" and "EXIT".

On hitting POST Button, XML fragment get generated and the same is being posted to Tally.ERP 9 running in a predefined port.

The VB Code for the same is:

```
Private Sub Command1_Click()
    Dim ComboString1 As String
    Dim ComboString2 As String
    Dim ComboString3 As String
    Dim ComboString11 As String
    Dim ComboString21 As String
    Dim ComboString31 As String
    ComboString11 = Combo1.Text
    ComboString21 = Combo2.Text
    ComboString31 = Text5.Text
    ComboString1 = Combo1.Text
    ComboString2 = Combo2.Text
    ComboString3 = Text5.Text
    If Combo1.Text = "" Or Combo2.Text = "" _
        Or Text1.Text = "" Or Text4.Text = "" Then
        MsgBox "Enters All The information", _
```

```

        vbApplicationModal, "Voucher Creation"

Else
    date2 = Format(Text1.Text, "dd/mm/yyyy")
    Temp  = Str$(Text4.Text * -1)

If InStrRev(ComboString11, "&") Then
    ComboString1 = Replace(ComboString11, "&", "" & "")
End If

If InStrRev(ComboString21, "&") Then
    ComboString2 = Replace(ComboString21, "&", "" & "")
End If

If InStrRev(ComboString31, "&") Then
    ComboString3 = Replace(ComboString31, "&", "" & "")
End If

xmlstc = "<ENVELOPE>" + vbCrLf & _
    "<HEADER>" + vbCrLf & _
        "<VERSION>1</VERSION>" + vbCrLf & _
        "<TALLYREQUEST>Import</TALLYREQUEST>" + vbCrLf & _
        "<TYPE>Data</TYPE>" + vbCrLf & _
        "<ID>Vouchers</ID>" + vbCrLf & _
    "</HEADER>" + vbCrLf & _
    "<BODY>" + vbCrLf & _
        "<DESC>" + vbCrLf & _
        "</DESC>" + vbCrLf & _
        "<DATA>" + vbCrLf & _
            "<TALLYMESSAGE >" + vbCrLf & _
            "<VOUCHER VCHTYPE=""Receipt"" ACTION=""Create"">" + vbCrLf & _
            "<DATE>" + date2 + "</DATE>" + vbCrLf & _
            "<NARRATION>" + ComboString3 + "</NARRATION>" + vbCrLf & _
            "<VOUCHERTYPENAME>Receipt</VOUCHERTYPENAME>" + vbCrLf & _
            "<EFFECTIVEDATE>" + date2 + "</EFFECTIVEDATE>" + vbCrLf & _
            "<ALLLEDGERENTRIES.LIST>" + vbCrLf & _
            "<LEDGERNAME>" + ComboString2 + "</LEDGERNAME>" + vbCrLf & _

```

```

        "<AMOUNT>" + Text4.Text + "</AMOUNT>" + vbCrLf & _
        "</ALLLEDGERENTRIES.LIST>" + vbCrLf & _
        "<ALLLEDGERENTRIES.LIST>" + vbCrLf
xmlstc = xmlstc + "<LEDGERNAME>" + ComboString1 +
        "</LEDGERNAME>" + vbCrLf & _
        "<ISDEEMEDPOSITIVE>Yes</ISDEEMEDPOSITIVE>" + vbCrLf & _
        "<AMOUNT>" + Temp + "</AMOUNT>" + vbCrLf & _
        "</ALLLEDGERENTRIES.LIST>" + vbCrLf & _
        "</VOUCHER>" + vbCrLf & _
        "</TALLYMESSAGE>" + vbCrLf & _
        "</DATA>" + vbCrLf & _
        "</BODY>" + vbCrLf & "</ENVELOPE>"

ServerHTTP.Open "POST", "http://localhost:" + PortNumber
ServerHTTP.send xmlstc

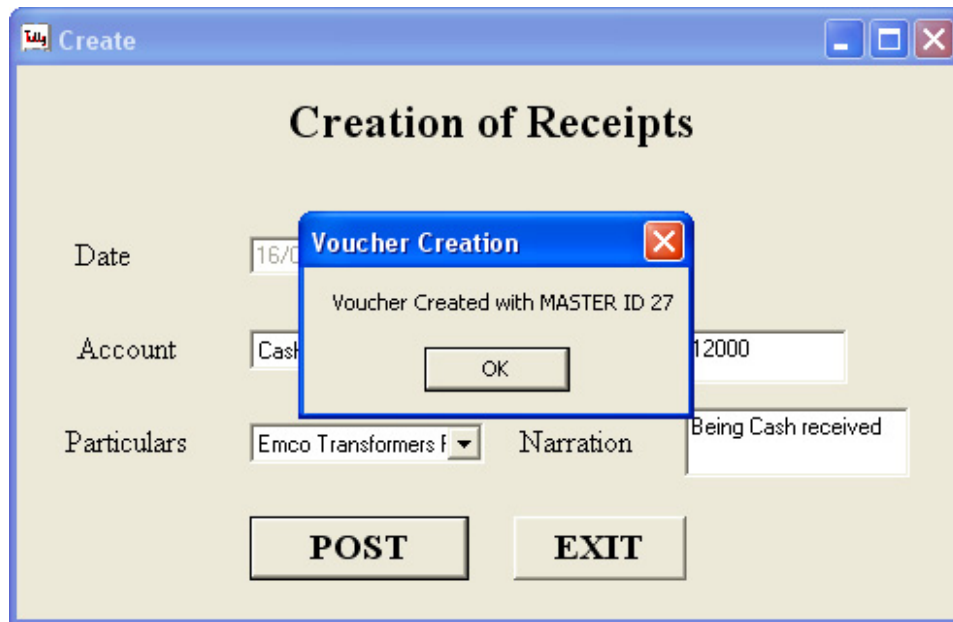
' Response from Tally - ServerHTTP.responseText
responsstr = ServerHTTP.responseText
newstring = InStrRev(responsstr, "<LINEERROR>")
If newstring = 0 Then]
    XMLDOM.loadXML (responsstr)
    MsgBox "Response String " + responsstr
    Set CHILDNODE = _
    XMLDOM.selectNodes("ENVELOPE/BODY/DATA/IMPORTRESULT/LASTVCHID")
    MsgBox "Voucher Created with MASTER ID " +
        CHILDNODE(0).Text, , "Voucher Creation"
Else
    MsgBox "Failed to POST"
End If
responsestr = ServerHTTP.responseText
Text4.Text = ""
Text5.Text = ""
End If
End Sub

```

Similar to Master Import, the following VB Code snippet sends the above generated XML Data to Tally.ERP 9 which is running at a predefined port.

```
ServerHTTP.Open "POST", "http://localhost:" + PortNumber 'for example:  
http://localhost:9000  
ServerHTTP.send xmlstc
```

On creating Vouchers in Tally.ERP 9, it sends the Response is parsed and Master ID is displayed as shown above



The above Screen displays the Receipt Entry created from the external Interface.

The screenshot displays the Tally.ERP 9 Accounting Voucher Alteration (Secondary) window. The window title is "Tally.ERP 9". The menu bar includes: P: Print, E: Export, M: E-Mail, O: Upload, L: Language, K: Keyboard, K: Control Centre, H: Support Centre, H: Help. The window is titled "Accounting Voucher Alteration (Secondary)" and "Global Enterprises". The voucher type is "Receipt" and the number is "No. 1". The date is "16-Feb-2009" and the day is "Monday". The account is "Cash" with a current balance of "12,000.00 Dr". The particulars are "Emco Transformers Pvt. Ltd." with a current balance of "12,000.00 Cr". The amount is "12,000.00". The narration is "Being Cash received". The window has a status bar at the bottom showing "Tally MAIN --> Gateway of Tally --> Display Menu --> Day Book --> Accounting Voucher: Alteration (Secondary)" and "(c) Tally Solutions Pvt. Ltd., 1988-2008 Mon, 16 Feb, 2009 16:17:08".

Particulars	Amount
Emco Transformers Pvt. Ltd.	12,000.00
	12,000.00

Narration:
Being Cash received

The above Interface is designed for Receipt Alteration based on Master ID of all the Vouchers. It lists the Master IDs of all the Receipt Vouchers in the List Box. On selection of a Master ID, it extracts all the Info pertaining to the selected Voucher as shown in the next figure.

The screenshot shows a software window titled "Alter" with a subtitle "Alteration of Receipts". The window contains several input fields and three buttons at the bottom.

Master ID	Alter ID	Voucher Number	Date
27	31	1	2/16/2009

Account	Cash	Amount	12000
Particulars	Emco Transformers P	Narration	Being Cash received

At the bottom of the window, there are three buttons: **ALTER**, **REFRESH**, and **EXIT**.

The above Interface displays the details of the Voucher pertaining to the selected Master ID. Only Alteration of Amount and Narration have been allowed in the external interface alteration.

Alteration of Receipts

Master ID	Alter ID	Voucher Number	Date
27	31	1	2/16/2009

Account	Cash	Amount	12600
Particulars	Emco Transformers P	Narration	Being Cash received

ALTER REFRESH EXIT

Amount has been altered from 12000 to 12600. On hitting Alter, XML Fragment will be generated and sent to Tally running at a predefined port.

XML generated for Altering the above is:

```
Private Sub Command1_Click ()
    Dim ComboString1 As String
    Dim ComboString2 As String
    Dim ComboString3 As String
    Dim ComboString11 As String
    Dim ComboString21 As String
    Dim ComboString31 As String
```

```

ComboString11 = Text2.Text
ComboString21 = Text3.Text
ComboString31 = Text5.Text
ComboString1 = Text2.Text
ComboString2 = Text3.Text
ComboString3 = Text5.Text

If InStrRev(ComboString11, "&") Then
    ComboString1 = Replace(ComboString11, "&", "" & "")
End If

If InStrRev(ComboString21, "&") Then
    ComboString2 = Replace(ComboString21, "&", "" & "")
End If

If InStrRev(ComboString31, "&") Then
    ComboString3 = Replace(ComboString31, "&", "" & "")
End If

date2 = Format(Text1.Text, "dd-mmm-yyyy")
If Text4.Text = "" Then
    MsgBox ("Please enter some value")
    Text4.SetFocus
Else
    Temp = Str$(Text4.Text * -1)
    xmlstc = _
        "<ENVELOPE>" + vbCrLf & _
        "<HEADER>" + vbCrLf & _
        "<VERSION>1</VERSION>" & _
        "<TALLYREQUEST>Import</TALLYREQUEST>" + vbCrLf & _
        "<TYPE>Data</TYPE>" + vbCrLf & _
        "<ID>Vouchers</ID>" + vbCrLf & _
        "</HEADER>" + vbCrLf & _
        "<BODY>" + vbCrLf & _
        "<DESC>" + vbCrLf & _
        "</DESC>" + vbCrLf & _
        "<DATA>" + vbCrLf & _

```

```

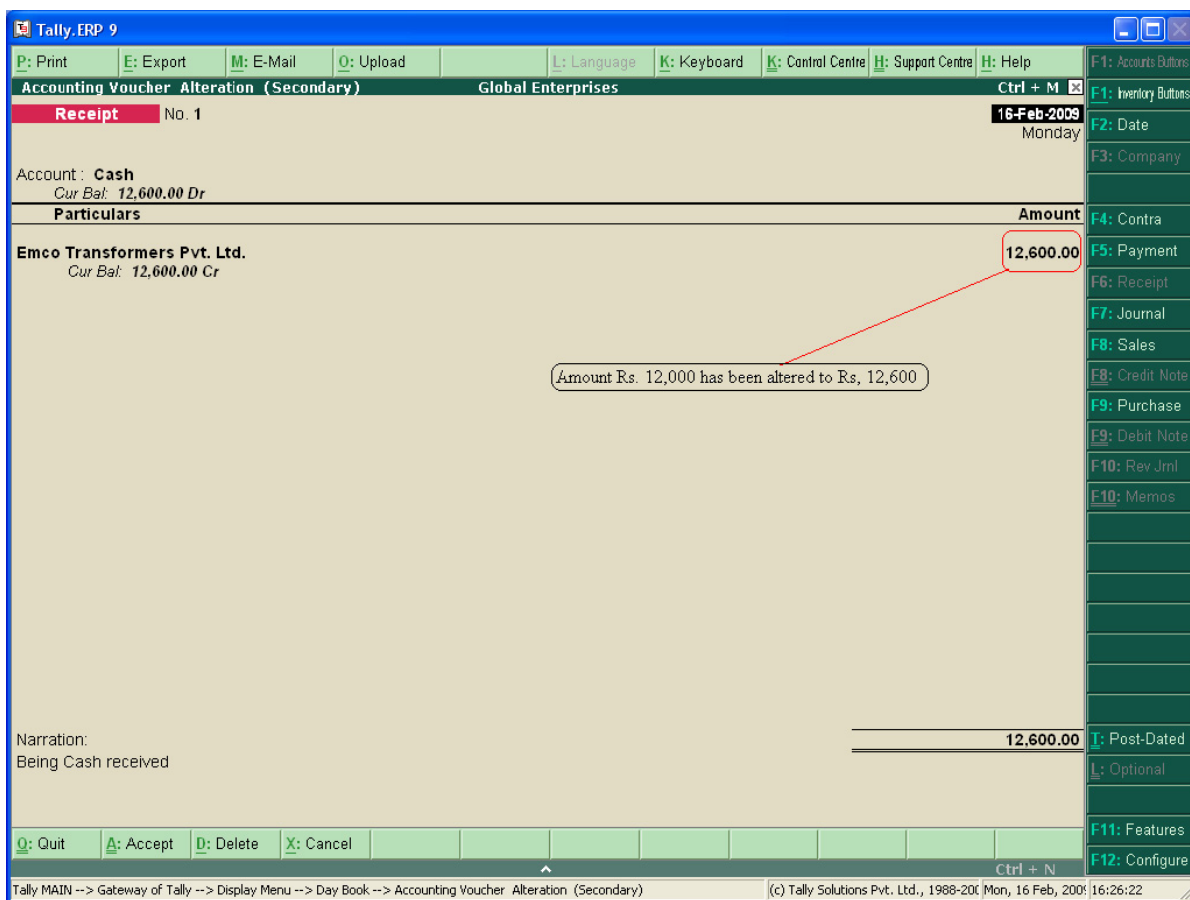
"<TALLYMESSAGE>" + vbCrLf & _
"<VOUCHER DATE=" + """" + date2 + """" +
    "TAGNAME=""MASTER ID"" TAGVALUE=" + """" +
    List1.Text + """" + " ACTION=""Alter""
    VCHTYPE = ""Receipt"">" + vbCrLf & _
xmlstc = xmlstc & _
"<ALLEDGERENTRIES.LIST>" + vbCrLf & _
"<LEDGERNAME>" + ComboString2 + "</LEDGERNAME>" + vbCrLf & _
"<AMOUNT>" + Text4.Text + "</AMOUNT>" + vbCrLf & _
"</ALLEDGERENTRIES.LIST>" + vbCrLf & _
"<ALLEDGERENTRIES.LIST>" + vbCrLf & _
"<LEDGERNAME>" + ComboString1 + "</LEDGERNAME>" + vbCrLf & _
"<ISDEEMEDPOSITIVE>Yes</ISDEEMEDPOSITIVE>" + vbCrLf & _
"<AMOUNT>" + Temp + "</AMOUNT>" + vbCrLf & _
"</ALLEDGERENTRIES.LIST>" + vbCrLf & _
"<NARRATION>" + ComboString3 + "</NARRATION>" + vbCrLf & _
"</VOUCHER>" + vbCrLf & _
"</TALLYMESSAGE>" + vbCrLf & _
"</DATA>" + vbCrLf & _
"</BODY>" + vbCrLf & _
"</ENVELOPE>"

ServerHTTP.Open "POST", "http://localhost:" + PortNumber
ServerHTTP.send xmlstc
responsstr = ServerHTTP.responseText
newstring = InStrRev(responsstr, "<LINEERROR>")
If newstring = 0 Then
    MsgBox "Save Successful", vbOKOnly, "Voucher : "
Else
    MsgBox responsstr
    MsgBox "Failed to POST"
End If
Text1.Text = ""
Text2.Text = ""
Text3.Text = ""
Text4.Text = ""

```

```
Text5.Text = ""
Text6.Text = ""
Text7.Text = ""
End If
End Sub
```

On executing the above code, the Tally Voucher containing the above selected Master ID gets altered with the given Amount and Narration details.



Tally.ERP 9

P: Print E: Export M: E-Mail O: Upload L: Language K: Keyboard K: Control Centre H: Support Centre H: Help

Accounting Voucher Alteration (Secondary) Global Enterprises Ctrl + M

Receipt No. 1 16-Feb-2009 Monday

Account: Cash Cur Bal: 12,600.00 Dr

Particulars	Amount
Emco Transformers Pvt. Ltd. Cur Bal: 12,600.00 Cr	12,600.00
Narration: Being Cash received	
	12,600.00

Amount Rs. 12,000 has been altered to Rs. 12,600

Q: Quit A: Accept D: Delete X: Cancel

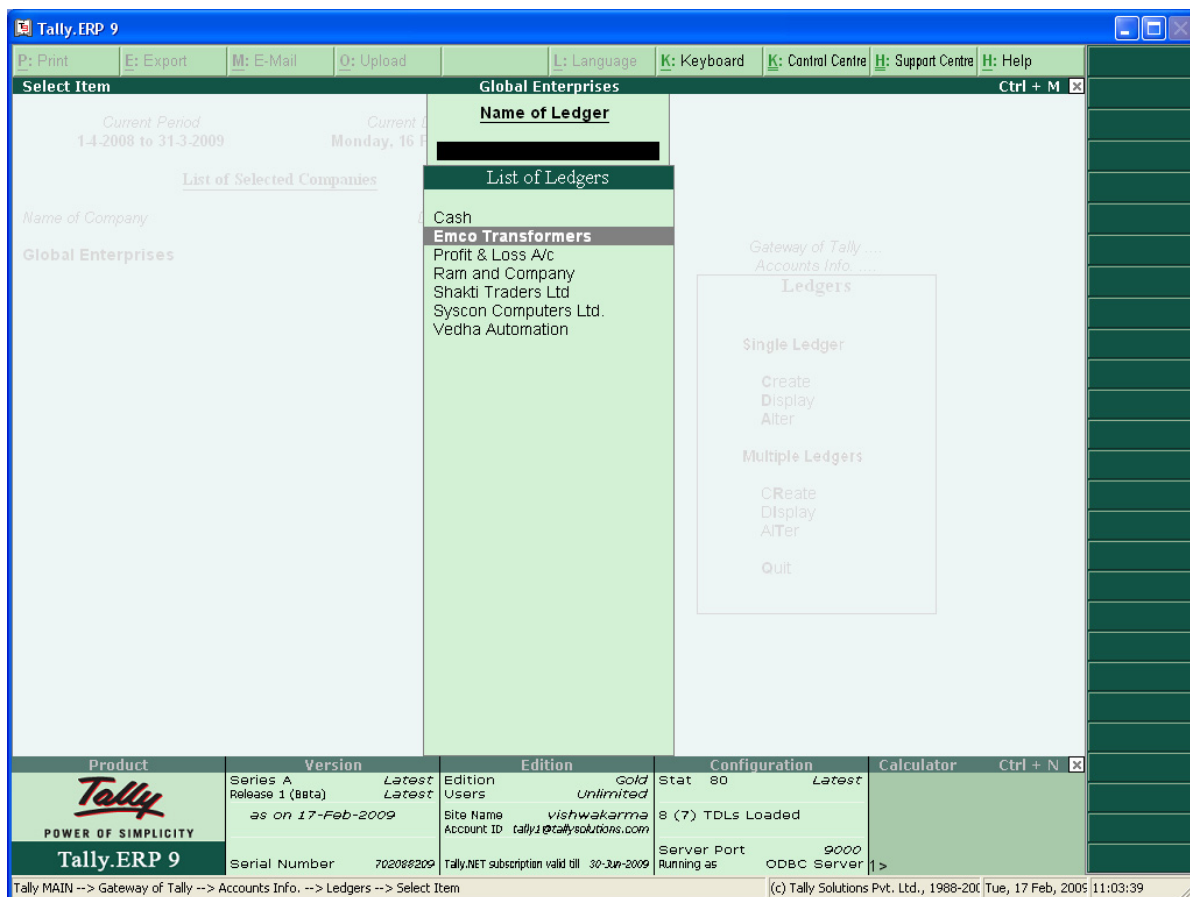
Tally MAIN --> Gateway of Tally --> Display Menu --> Day Book --> Accounting Voucher Alteration (Secondary) (c) Tally Solutions Pvt. Ltd., 1988-2001 Mon, 16 Feb, 2009 16:26:22

The above Tally.ERP 9 Screen displays the Voucher which has been altered from an external interface application.

2.1.3 Case Study III – Exporting Ledger Masters from Tally.ERP 9 to External Application

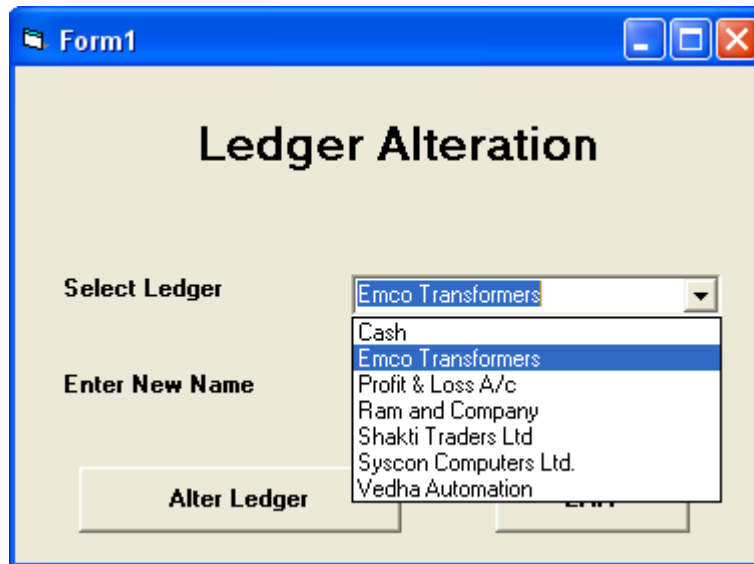
A Company Global Enterprises needs to design an interface for displaying list of ledgers in VB and allowing the user to alter the same through external interface. Finally the altered ledger must be posted to Tally.ERP 9.

Ensure that the TDL Ledger Report.txt has been associated in Tally.ERP 9. This TDL report is written for displaying List of Ledgers with the required XML Tags since when an export request is sent to Tally.ERP 9, these XML Tags can be located and displayed in the Interface.



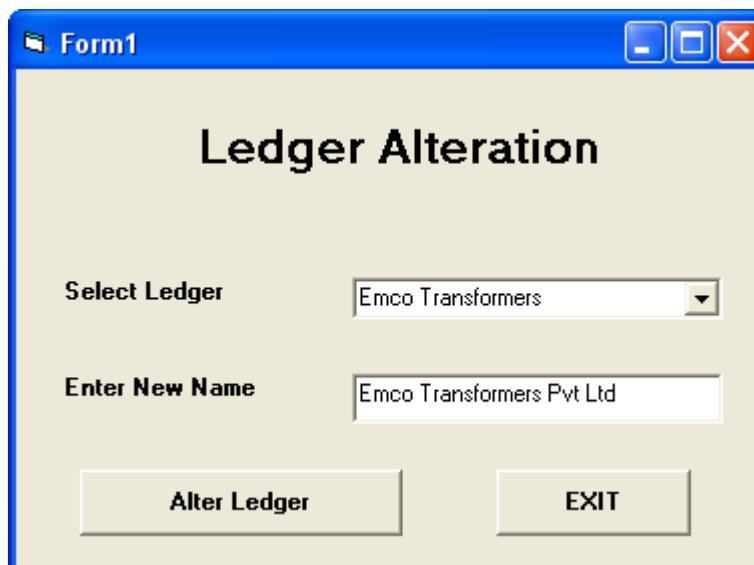
The above Tally.ERP 9 Screen displays the List of Ledgers prior to alteration in Company Global Enterprises.

The following interface has been designed in VB for Ledger Alteration. The Ledgers in Global Enterprises Company has been gathered through XML and displayed in drop down list for user selection.



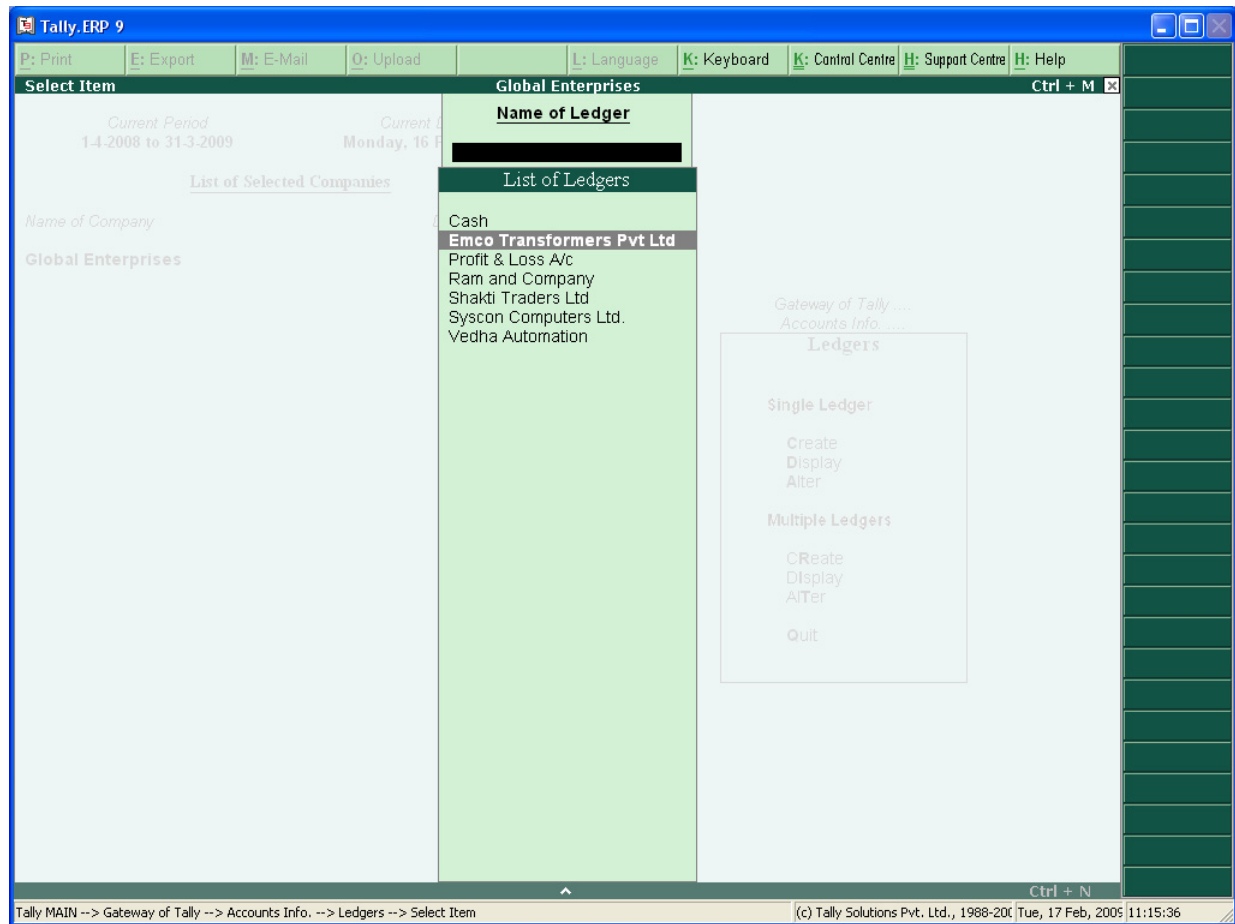
The screenshot shows a window titled 'Form1' with the heading 'Ledger Alteration'. It contains two labels: 'Select Ledger' and 'Enter New Name'. The 'Select Ledger' dropdown menu is open, displaying a list of ledger names: 'Cash', 'Emco Transformers', 'Profit & Loss A/c', 'Ram and Company', 'Shakti Traders Ltd', 'Syscon Computers Ltd.', and 'Vedha Automation'. The 'Emco Transformers' option is currently selected. Below the labels are two buttons: 'Alter Ledger' and 'EXIT'.

On selection of the Ledger, an additional input field Enter New Name allows the user to give a different name to the selected ledger.



The screenshot shows the same 'Form1' window with the heading 'Ledger Alteration'. The 'Select Ledger' dropdown menu is now closed, and 'Emco Transformers' is visible in the text field. The 'Enter New Name' text field now contains the text 'Emco Transformers Pvt Ltd'. The 'Alter Ledger' and 'EXIT' buttons remain at the bottom.

On hitting Alter Ledger button, the XML for Ledger Alteration is generated and posted to Tally.ERP 9 and the Tally.ERP 9 Ledger is altered accordingly as shown in the following Tally.ERP 9 screen.



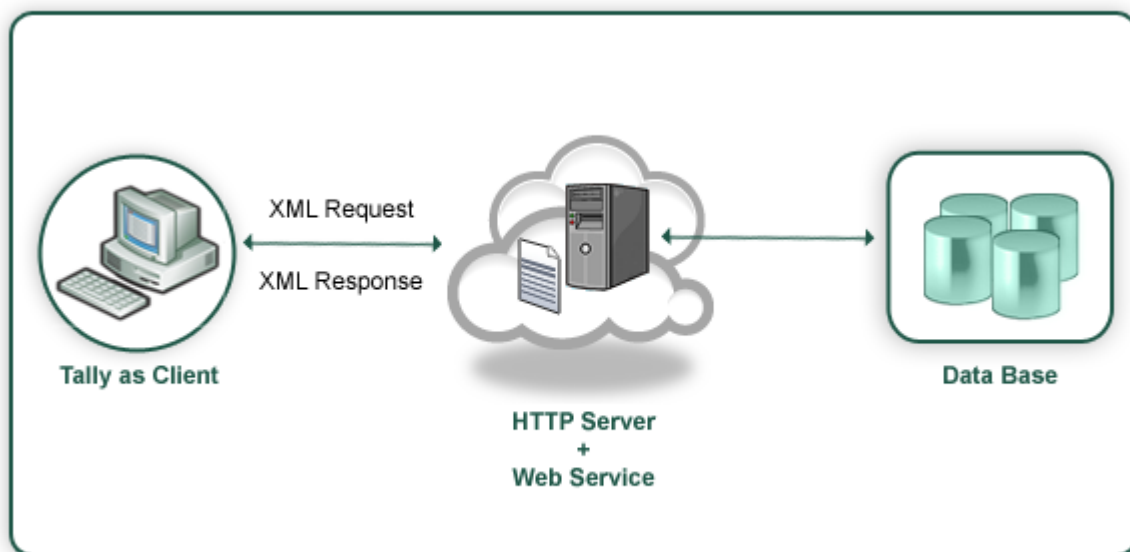
The above screen displays that the Ledger Emco Transformers has been renamed to Emco Transformers Pvt Ltd through external interface application.

2.2 Tally.ERP 9 as a Client – Tally as a Front end for Web Services

2.2.1 Introduction

With the new enhancements now Tally can send a request to the HTTP Server and display the response. Tally sends the request as an XML and receives the response in XML format. Tally can collect data directly from an XML file or from a web service running on the server. Even data can be sent along with the request to the web service which can process the data and return the result as an XML. The web service in turn might be operating on an external data base.

Following Figure shows the functionality when Tally acting as a client interacts with a web service operating on the external data.



Collection capability has been enhanced to gather live data from HTTP/web-service delivering XML. The entire XML is now automatically converted to TDL objects and is available natively in TDL reports as \$ based methods. Reports can be shown in Tally with live data from an HTTP server. Coupled with the new [OBJECT:] extensions and Using POST action data can be submitted data back to the server almost operating Tally as a client to HTTP-XML web-services.

This capability allows us to retrieve and store data as objects in Collection. The attributes in collection for gathering XML based data from a remote server over HTTP are RemoteURL, RemoteRequest, XMLObjectPath, and XMLObject. Whenever the collection is referred the data is fetched from the remote server and is populated in the collection.

Events On Focus, On Form Accept and Action - HTTP - POST are used providing the flexibility to send a request automatically to server when a particular event occurs.

Consider that the file TestXML.xml contains data and Tally sends a request to fetch the data. Once the data is available any data manipulation operation can be performed on it.

The TestXML.xml file contains the following:

```
<CUSTOMER>
  <NAME>Sapna Awasthi</NAME>
  <EMPID>1000</EMPID>
  <PHONE>
    <OFFICENO>080-66282559</OFFICENO>
    <HOMENO>011-22222222</HOMENO>
    <MOBILE>990201234</MOBILE>
  </PHONE>
  <ADDRESS>
    <ADDRLINE>C/o. Info Solutions</ADDRLINE>
    <ADDRLINE>Technology Street</ADDRLINE>
    <ADDRLINE>Tech Info Park</ADDRLINE>
  </ADDRESS>
</CUSTOMER>
```



Prerequisites for Data Transfer over HTTP

In order to retrieve the data available in TestXML.xml File from a remote server (Pre-defined IP Address) ensure that web service is running on the machine. Check for IIS Server Installation. The file TestXML.xml can be copied to the directory C:\inetpub\wwwroot to be accessible at the root and then the URL can be specified as follows <http://localhost/TestXML.xml>.

If the XML request needs to be processed at the remote server by a file (.asp, .php, etc.), at least one web server (e.g., IIS, Apache etc) and PHP/ASP must be installed on the system.

Collection attributes introduced to facilitate these capabilities are:

```
[Collection: <Collection Name>]
  RemoteURL      : <http-url formula>
  RemoteRequest  : <Request-report-name>,+
                  <pre-request-display-report>: +
                  <Encoding type>
  XMLObjectPath  : <Start-node> : <Path-to-start-node>
  XMLObject      : <TDL-Object-Name>
```

Once the collection is populated with the objects it can be used in TDL reports to display the retrieved data. The XML TAG names can be used as methods in the TDL programs.

The methods names must be in the same case as the XML Tag name.

Example :

- If the XML tag name is written as <NAME> then \$NAME must be used as method name while referring it in TDL.
- If the XML tag name is written as <Name> then \$Name must be used as method name while referring it in TDL.

2.2.2 Collection attribute – Remote URL

Remote-URL attribute is used to specify the Universal Resource Locator (URL) of the HTTP server delivering the XML data. The URL can be directly specified as string or through a String formula. This attribute is mandatory to access the data in XML format from remote server in a collection.

The collection is created as follows to populate XML Data available at the URL <http://localhost/TestXML.xml> :

Syntax

RemoteURL : <http-url formula>

Where,

<http-url formula> This can be any string formula which resolves as an URL.

Example :

```
[Collection: XML Get Collection]
    Remote URL    : "http://localhost/TestXML.xml"
```

This collection can be used in a TDL Report to display the data retrieved. In order to display the retrieved data the line is repeated over this collection.

```
Repeat: XMLDetLn : XMLGetCollection
```

In the field the XML TAG names are used as method names.

```
[Field : CustNm]
    Set As : $NAME
```

To display the PHONE and ADDRESS details, the line can be exploded.

```
[Line : XMLDetLn]
    Explode : PhPart
    Explode : AddPart
```

n the Exploded parts the line is repeated over the PHONE and ADDRESS collection respectively.

```
[Part : PhPart]
    Repeat : PhLine : PHONE
[Part: AddPart]
    Repeat: AddLine : ADDRESS
```

2.2.3 Collection attribute – XML Object Path

By default, all the data from XML file is made available in the collection. If only a specific data fragment is required it can be obtained using the collection attribute XML Object Path. This attributes converts the extracted fragment as TDL Objects in Collection. By default, it takes the root node.

Syntax

XMLObjectPath : <Start-node> : <Path-to-start-node>

Where,

<Start-Node> allows you to specify the name and position of the XML node from which the data should be extracted. The parameter is specified as follows:

<Node Name> : <Position>

<Path-to-Start-Node> is used to specify the path to reach the <start node> from the root node.

The path specification is:

<Root-node>: <Child Node> : <Start Pos> : <Child Node> : <Start Pos> ...

Example :

From the XML file, if only address is required then the collection is defined as follows:

```
[Collection: XML Get CollObjPath]
    Remote URL      : "http://localhost/TestXML.xml"
    XML Object Path : ADDRESS: 1: CUSTOMER
```

2.2.4 Collection attribute – XML Object

XMLObject attribute is used to specify the TDL Object specification.

Syntax

XMLObject : <TDL-Object-Name>

Where,

<TDL -Object Name> is user defined name. The data is required to be mapped as TDL Objects. The following syntax is used for object specification:

```
[Object: <Object Name>]
    Storage      : <Name>      : Type
    Collection   : <Name>      : Type
```

The second Parameter in the Collection Type can be an Object type in case of a complex collection or a simple data type in case of simple collection.

Example :

The data available in XML format is at the URL "http://Remoteserver/TestXML.xml". The collection attribute XML Object is used to specify the object name to which the obtained data is mapped.

```
[Collection: XML Get Collection]
    Remote URL      : "http://Remoteserver/TestXML.xml"
    XML Object      : Customer Data
```

The Object specification for "Customer Data" is as follows:

```
[Object: Customer Data]
    Storage      : Name      : String
    Storage      : EmpId     : String
    Collection    : Phone     : XML Phone Coll      ;; Complex Collection
    Collection    : ADDRESS   : XML AddressColl     ;; Complex Collection

[Object: XML Phone Coll]
    Storage      : OfficeNo   : String
    Storage      : HomeNo     : String
    Storage      : Mobile     : String

[Object: XML AddressColl]
    Collection    : AddrLine   : String              ;; Simple collection
```

2.2.5 Collection attribute – Remote Request

A TDL report can be sent to the HTTP server as an XML request and the XML response is to be obtained in the collection. RemoteRequest attribute is used to specify the Report name which is to be sent to the HTTP server as an XML Request. If the report requires user inputs then it has to be accepted before the request is sent. Pre-request is used to specify the name of the report which accepts the user-input.

Syntax

RemoteRequest: <request-report-name>,<pre-request-report>: +
<encoding type>

Where,

<Encoding Type> specifies the encoding to be used while transmitting information to the receiving end. The valid encoding formats are ASCII and UNICODE. UNICODE is set by default.

<Request Report Name> is the name of the TDL Report which will be used for generating XML Request to be sent.

<Pre-Request Report Name> is the name of the TDL Report which accepts the user-input.

Example :

The Test.php page on the remote server accepts the data in the following XML format.

```
<ENVELOPE>
  <REQUEST>
    <NAME>Tally</NAME>
    <EMPID>00000</EMPID>
  </REQUEST>
</ENVELOPE>
```

Following collection sends request in the above XML format with the help of a TDL report XML-PostReqRep. The encoding scheme selected is ASCII.

```
[Collection: XML Post Collection]
Remote URL      : "http://Remoteserver/test.php"
RemoteRequest   : XMLPostReqRep : ASCII
```

The report XMLPostReqRep is automatically executed when the collection is referred.

The Request Report

In the request report XMLPostReqRep, the XMLTAG attribute is used at Part and Field Definitions. The XML Tag <Envelope> is added by Tally while sending the XML request.

In the part definition the XMI TAG is specified as "REQUEST".

```
[Part: XMLPostReqRep]
XML Tag : "REQUEST"
Scroll  : Vertical
```

In the field definitions the tag names "NAME" and "EMPID" are specified.

```
[Field: XMLPostReqRepName]
```

```
XML Tag : "NAME"
```

```
Set As : " Tally "
```

```
[Field: XMLPostReqRepId]
```

```
XML Tag : " EMPID "
```

```
Set As : " 00000 "
```

The response received from "http://Remoteserver/test.php" page is the same XML given previously. The data now available in the collection can be displayed in a TDL report.

Pre- request Report

A Pre Request Report is required when some inputs are to be accepted from the user and the XML Request is to be generated out of those inputs. In that case, a TDL report is used which has to be accepted first. If the data captured through pre request report has to be sent to remote server for processing then it has to be made available in the Request Report. The input report name is specified as Pre Request Report.

```
[Collection: XML Post Collection]
```

```
Remote URL : "http://localhost/test.php"
```

```
RemoteRequest : XMLPostReqRep, XML PreReqRep : ASCII
```

The Report XMLPostReqRep sends the XML request to the page Test.php in the format described earlier. Before sending the XML request to the page, the data entered in the report XML PreReqRep must be accepted. The data entered in the Pre-Request report can also be sent to the remote server in the XML request. Both the reports are triggered when the collection is referred. PreRequest Report is only used to accept value from the user. These values are sent to the HTTP Server through the Request Report Only.

2.2.6 Action – HTTP POST

A new Key/ Button Action HTTP Post has been introduced which will help in exchanging data with external applications using web services. In other words, HTTP Post Action can be used to submit data to a server over HTTP and gather the response. This will enable a TDL Report to perform a HTTP Post to a remote location.

Syntax

```
[Key: <Key Name>]
```

```
Key : <Key Combination>
```

```
Action : HTTP Post : <URL Formula> : <Encoding> : <Request Report>: +
```

```
<Error Report> : <Success Report >
```


Where,

<URL Formula> can be any string formula which resolves as an URL and is defined under System Definition.

<Encoding> is the encoding scheme ASCII or UNICODE .

<Request Report> is the name of the TDL Report which will be used for generating XML Request to be sent.

<Error Report > is displayed in case of failure.

<Success Report> is displayed when the post is successful.

The details pertaining to URL (at the receiving end), Encoding Format, RequestReport , Error Report and Success Report should be specified along with HTTP Post Action. The Request, Error and Success reports are optional.

Success and failure is determined by <STATUS> tag in the standard message format. If it is 1 then success else it is failure. Based on the value of the <STATUS> tag 0/1, the error report and success report are executed respectively. It will not close or accept the form if status is not equal to 1.

Example :

```
[Key: XMLReqResp]
    Key      : Ctrl + R
    Action   : HTTP Post : @@MyUrl : ASCII : ReqRep: ERRRespRep: SuccRep
    Scope    : Selected Lines
;;URL Specification must be done as a system formula
[System: Formula]
    MyUrl    : http://127.0.0.1:9000
```

The defined Key XMLReqResp in the snippet above must be attached to an initial Report. When the report is activated and this Key is pressed, the Action HTTP Post activates a defined report ReqRep which generates the request XML. The response data is made available in collection called Parameter Collection. The reports ERRRespRep and SuccRep can use the Parameter Collection to display the error message/data in the Report.

Two events are introduced in Tally.ERP 9, they are:

- OnFormAccept
- OnFocus

2.2.7 Event – On Form Accept

A new event is introduced in Form definition, On: Form Accept. A list of actions can be executed when the form is accepted. The execution can be based on a condition.

Syntax

```
On: Form Accept: <Condition>: Action: Action parameters
```

Where,

<Condition> should return a logical value.

<Action> any one of the actions

<Action Parameters> parameters of the actions specified.

Example :

```
[Form : TestForm]
On:FormAccept:Yes:HttpPost:@@SCURL:ASCII:SCPostNewIssue:SC NewIssueResp
```

2.2.8 Event – On Focus

In the definitions Part, Line and Field a new event **On : Focus** is introduced. When the Part, Line or Field receives a focus list of action can be executed. A condition can also be specified. When the condition returns true then only the actions will be executed.

Syntax

On : Focus : condition : Action : Action parameters

<Condition> should return a logical value

<Action> any one of the actions

<Action Parameters> parameters of the actions specified

Its a list type attribute so as many actions can be specified.

```
[Part : TestPart1]
On : FOCUS : Yes : HTTP Post : @@MyUrl : ASCII : ReqRep, RespRep
[Part : TestPart2]
On : FOCUS : Yes : Call : SCSetVariables : $$Line
```

2.3 Collection Capability to Accept File as a Data Source

Collection is enhanced to support a generalized data source structure. All types of data sources are currently made available through different attributes of collection. For example: To retrieve data from ODBC collection attributes, ODBC, SQL, and SQL Object are used.

The new attribute Data Source can be used for any type of Data source.

2.3.1 The Collection attribute – Data Source

As of now the attribute data source allows to specify XML file as data source. The collection can be created directly from the specified XML file and the data in the XML file can be displayed in a report.

Syntax

DataSource : <Type> : <Encoding> : <Identity>

<Type> specifies the type of data source. Eg. File Xml, HTTP XML, ODBC etc

<Encoding> ASCII or UNICODE. This is Optional .The default value is UNICODE.

<Identity> Data source file path

Example :

```
[Collection: My XML Coll]
DataSource : File Xml : "C:\MyFile.xml"
```

In the above code snippet the type of file is 'File XML ' as the data source is XML file. The encoding is Unicode by default as it is not specified.



Support for other data sources like ODBC, HTTP will also be available in future builds.

2.4 Case Study

A web service running on the remote server is used to extract data from the external data base. It sends the response in the following XML format based on the data available in the database.

```
<ENVELOPE>
  <HEADER>
    <VERSION>version</VERSION>
    <TALLYREQUEST>Import</TALLYREQUEST>
    <TYPE>DATA</TYPE>
    <STATUS>1</STATUS>
  </HEADER>
  <BODY>
    <DATA>
      <STKITNAME>
        <ITNAME>Item 1</ITNAME>
        <STKBAT>
          <BATNAME>B1</BATNAME>
          <BATQTY>100</BATQTY>
          <BATRATE>10</BATRATE>
          <BATAMT>1000</BATAMT>
        </STKBAT>
      </STKBAT>
    </DATA>
  </BODY>
</ENVELOPE>
```

```

        <BATNAME>B2</BATNAME>
        <BATQTY>10</BATQTY>
        <BATRATE>10</BATRATE>
        <BATAMT>100</BATAMT>
    </STKBAT>
</STKITNAME>
<STKITNAME>
    <ITNAME>Item 2</ITNAME>
    <STKBAT>
        <BATNAME>B3</BATNAME>
        <BATQTY>100</BATQTY>
        <BATRATE>10</BATRATE>
        <BATAMT>1000</BATAMT>
    </STKBAT>
    <STKBAT>
        <BATNAME>B4</BATNAME>
        <BATQTY>10</BATQTY>
        <BATRATE>10</BATRATE>
        <BATAMT>100</BATAMT>
    </STKBAT>
</STKITNAME>
</DATA>
</BODY>
</ENVELOPE>

```

Tally can use the web service to fetch data from the data base and perform following operation:

- ❑ Display the retrieved data in Tally as a Report
- ❑ Retrieve data based on the request sent by Tally
- ❑ Send the request based on user input to the web server and display the response in a report

Display the retrieved data in Tally as a Report:

To retrieve the data from the database the STKXML.php page is used and this page sends the response to Tally in the XML format mentioned above.

To display the data as a report in Tally, a TDL report needs to be designed. A line in the report repeats over a collection of objects obtained from the XML. As the header information is also available in the XML, XML Object Path attributes must be used to specify the node from which the data should be retrieved in collection.

A collection is created as follows:

```
[Collection : STKItColl]
    Remote URL      : "http://remoteserver/STKXML.php"
    XML Object Path : STKITNAME:1:ENVELOPE:BODY:1:DATA:1
```

In the part definition a line is repeated over the collection STKItColl as shown:

```
[Part : StkItDisp]
    Line   : StkItDisp
    Repeat : StkItDisp : STKItColl
```

In this line only the Name can be displayed as there are multiple batches the line should be exploded to display the Batch Details.

```
[Line : StkItDisp]
    Field   : StkItDispNm
    Explode : StkItDispBat
```

In the exploded part the line must be repeated over the collection StkBat in order to display the batch details.

```
[Part :StkItDispBat]
    Line   : StkItDispBat
    Repeat: StkItDispBat :STKBAT
```

Data Retrieval based on the request sent by Tally:

In the scenario when the data is to be retrieved based on the request sent by Tally.ERP 9, Request report should be designed. Consider that this report generates the following XML format which is sent to the web server running the php page:

```
<ENVELOPE>
    <HEADER>
        <VERSION>version</VERSION>
        <TALLYREQUEST>Export</TALLYREQUEST>
        <TYPE>DATA</TYPE>
    </HEADER>
```

```
<BODY>
    <DATA>
        <SVSTKITEM> Item 1 </SVSTKITEM>
    </DATA>
</BODY>
</ENVELOPE>
```

The Item name is sent along with the request and all the batches of this item should be retrieved from the data base.

The collection is defined as follows:

```
[Collection : STKItColl]
    Remote URL      : "http://remoteserver/STKXML.php"
    XML Object Path : STKITNAME:1:ENVELOPE:BODY:1:DATA:1
    Remote Request  : StkItReq :ASCII
```

The report "StkItReq" generates the XML format which is sent from Tally to the web service. The Request report "StkItReq" must contain two parts to provide the Header and Body tag information.

```
[From : StkItReq]
    Part : StkItReqh, StkItReqb
```

The parts will have the tags "HEADER" and "BODY" respectively. The attribute Scroll : Vertical must be specified for in both the parts.

```
[Part : StkItReqh]
    XML TAG : "HEADER"
```

The three fields in the line will have tags "VERSION", "TALLYREQUEST" and "TYPE" respectively. These are hard coded in this scenario.

```
[Part : StkItReqb]
    XML TAG : "BODY"
```

The Line in the part StkItReqb will have the XML tag as "DATA"

```
[Line : StkItReqbLn]
    Field : StkItReqbFld
    XML TAG: DATA
```

The field contains the XML tag as "SVStkItem". The Item name passed as the value of the field in the request sent as XML.

```
[Field: StkItReqbfld]
Set As : ##SVStkItem
XML TAG: SVSTKITEM
```

The following response is received from the web service

```
<ENVELOPE>
  <HEADER>
    <VERSION>version</VERSION>
    <TALLYREQUEST>Import</TALLYREQUEST>
    <TYPE>DATA</TYPE>
    <STATUS>1</STATUS>
  </HEADER>
  <BODY>
    <DATA>
      <STKITNAME>
        <ITNAME>Item 1</ITNAME>
        <STKBAT>
          <BATNAME>B1</BATNAME>
          <BATQTY>100</BATQTY>
          <BATRATE>10</BATRATE>
          <BATAMT>1000</BATAMT>
        </STKBAT>
        <STKBAT>
          <BATNAME>B2</BATNAME>
          <BATQTY>10</BATQTY>
          <BATRATE>10</BATRATE>
          <BATAMT>100</BATAMT>
        </STKBAT>
      </STKITNAME>
    </DATA>
  </BODY>
</ENVELOPE>
```

Data Retrieval based on the user input sent as request to the web server:

The data to be sent in the request can be accepted from the user. User can enter the name of the Item in the Pre Request Report. The values entered in the fields are passed through global variables to the request report. The Collection is defined as follows:

Example :

```
[Collection : STKItColl]
    Remote URL      : "http://remoteserver/STKXML.php"
    XML Object Path : STKITNAME:1:ENVELOPE:BODY:1:DATA:1
    Remote Request  : StkItReq, StkItPreReqRep :ASCII
```

The Pre Request Report "StkItPreReqRep" is created as a simple report which accepts the name of Stock Item from the user and stores the value in a global variable.

```
[Field : StkPreReqFld]
    Modifies      : SVStkItem
[System: Variables]
    SVStkItem     : ""
[Variable: SVStkItem]
    Type          : String
```

In the Request Report, the variables value is set in a field as follows:

```
[Field : StkItReqbFld]
    Set As : ##SVStkItem
    XML TAG: SVSTKITEM
```

Data Storage in External Database using data input in Tally

Consider the scenario when the batch details corresponding to a particular Stock Item are entered by the user in a report in Tally. This data needs to be sent to the Web Service which interacts and stores data in the External Database.

The data can be sent to the web service through a Request Report in two ways:

- Using the Request Report as a value for collection attribute Remote Request and referring to the collection
- Using the Request Report as parameter for action HTTP Post through a button

Specifying Request Report in Collection

In Tally a report should be designed to send the request to the web service using the collection attribute Remote Request.

The collection is created as follows:

```
[Collection : STKItColl]
    Remote URL      : "http://remoteserver/STKXML.php"
    XML Object Path  : STKITNAME: 1: ENVELOPE: BODY: 1: DATA: 1
    Remote Request   : StkItReq, StkPreReq :ASCII
```

The report "StkItReq" generates the XML format which is sent from Tally to the web service. The pre-request report "StkPreReq" is executed first and the data entered by the user is passed to the request report "StkItReq". The request report "StkItReq" generates the XML and send s it to the web server in following format.

```
<ENVELOPE>
    <HEADER>
        <VERSION>version</VERSION>
        <TALLYREQUEST>Export</TALLYREQUEST>
        <TYPE>DATA</TYPE>
    </HEADER>
    <BODY>
        <DATA>
            <SVSTKITEM>Item 1</ SVSTKITEM >
            <BATNAME>B3</BATNAME>
            <BATQTY>75</BATQTY>
            <BATRATE>100</BATRATE>
            <BATAMT>7500</BATAMT>
        </DATA>
    </BODY>
</ENVELOPE>
```

The request report is created in the similar fashion as explained in the previous section. The web service is programmed to store the data in the database.

Specifying Request Report with HTTP Post

The button can be added to an existing report which when clicked will execute the HTTP Post action.

In case of action HTTP Post the response received is available in the collection "Parameter Collection". Based on the Success or Failure of the action appropriate message should be displayed in the Error Report and Success Reports.

The Button is defined as explained in the section "**Action HTTP Post**".

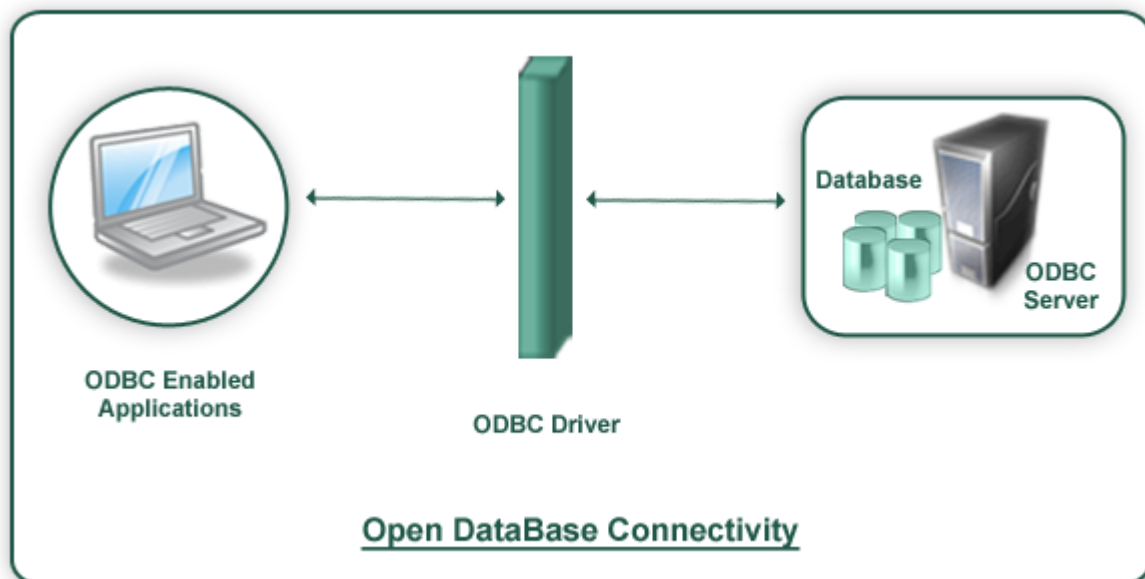
Lesson 3: Integration using ODBC Interface

On the completion of this chapter you will be able to

- Understand the Tally.ERP 9 – ODBC Interface
- Understand the functionality of Retrieving Data from External Database

3.1 Tally.ERP 9 – ODBC Interface

Open Database Connectivity (ODBC) is an interface for accessing data in a heterogeneous environment of relational and non- relational database management systems.



ODBC is an Application Program Interface (API) specification that allows applications to access multiple database systems using Structured Query Language (SQL). ODBC provides maximum interoperability-a single application can access many different database systems. This allows an ODBC developer to develop an application, without targeting a specific type of data source.

A typical ODBC implementation will have following components

- ODBC Client
- ODBC Driver
- ODBC Server

ODBC Client

An ODBC client implements ODBC API. The ODBC API in turn will communicate with the ODBC Driver provided by the Database.

ODBC Driver

The ODBC driver is a library that implements the functions supported by the ODBC API. It processes ODBC function calls, submits SQL requests to Database, and returns results back to the application.

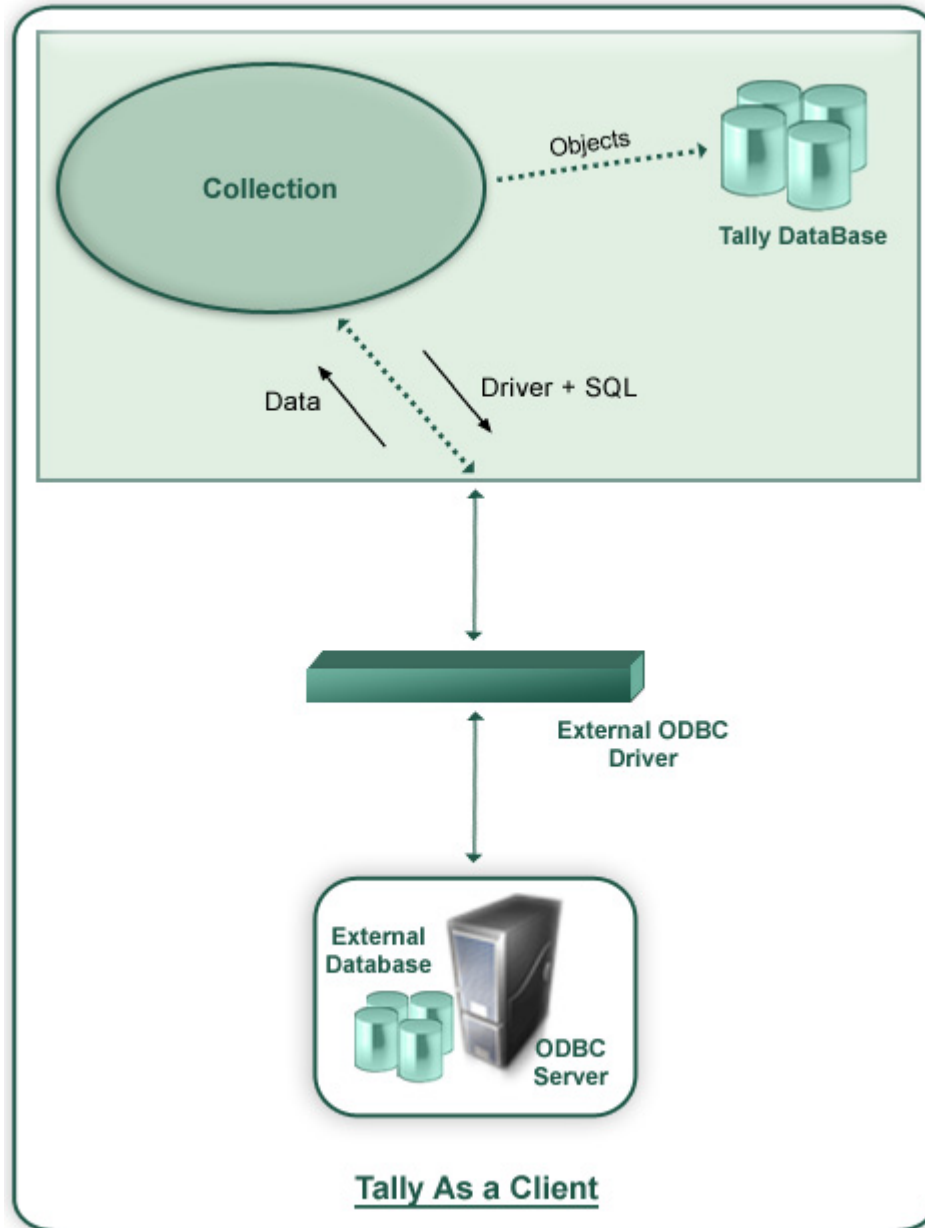
ODBC Server

A database which supports ODBC can understand the SQL. Normally ODBC Driver submits SQL request from the ODBC client and these SQL request will be executed and result will be given back to the ODBC client

Tally.ERP 9 is a application which can act as ODBC Server as well as ODBC Client

If any Application supports ODBC interface, then it needs to implement the ODBC API. The ODBC API in turn will communicate with the ODBC Driver provided by the Database. The ODBC driver is a library that implements the functions supported by the ODBC API. It processes ODBC function calls, submits SQL requests to Database, and returns results back to the application. Tally.ERP 9 can act as Application i.e. as ODBC client as well as Database i.e. ODBC server.

3.2 Tally.ERP 9 as a Client – Retrieving Data from External Database



Tally.ERP 9 is an ODBC enabled application. It can talk to ODBC Drivers of any external Database. In TDL, Collection is a definition which holds the data. Collection Definition has a capability to gather the data from the external data source through ODBC.

A Collection definition can communicate to the ODBC drivers of the external database either through Data Source Name (DSN) of the external database or through DSN less i.e. directly by mentioning ODBC Driver, Drive ID, path of the source, etc. In Collection Definition, SQL queries are used to gather the required information the external database.

Once required data is brought in to the Tally.ERP 9 application, the each row is treated as one Object and each column of that row as method of that Object. Thus the external data can be utilized inside the application.

Syntax

```
[Collection:<Collection Name>]
    ODBC : <Driver Info>
    SQL  : <SQL Statement>
```

Where **<Driver Info>** can be a DSN or ODBC driver, Driver ID & path of the data source can be mentioned and **<SQL Statement>** is SELECT query.

Example 1: Import the Ledger Master from MS Access

Sample Data

Ledger Master		
Led Name	Led Parent	Opening Balance
Ram & Shyam Party	Sundry Debtors	-101
Universal Agencies	Sundry Creditors	5000
Paramount Ltd	Sundry Debtors	1000
Bangalore Cable Network	Sundry Creditors	1000

3.2.1 TDL Collection to gather data from MS Access

```
[Collection: Led Coll From Access]
    ODBC : "Driver={Microsoft Access Driver +
           (*.mdb)};Dbq=C:\Masters.mdb;Uid=;Pwd=;"
    SQL  : Select * From LedgerMaster
```

```
[Collection: Led Coll]
    Source Collection : Led coll From Access
    Compute          : LedName   : $_1
    Compute          : LedParent : $_2
    Compute          : LedOpBal  : $$AsAmount:$_3
```

Alternatively TDL Collection gathers data from MS Access in following way

```
[Collection: Led Coll From Access]

  ODBC          : "Driver={Microsoft Access Driver +
                  (*.mdb) };Dbq=C:\Masters.mdb;Uid=;Pwd=;"

  SQL           : Select * From LedgerMaster

  SQL Object    : AccessObj

[Object: AccessObj]

  LedName       : $_1
  LedParent     : $_2
  LedOpBal      : $$AsAmount:$_3
```

Utilizing the ODBC Collection in a user defined Function to store the Ledger Objects in Tally DB

```
[Function: Ledger Import]

  01 : WALK COLLECTION: Led Coll
  02 : IF : @@LedgerExists > 0
  03 : NEW OBJECT: Ledger : $LedName
  04 : CALL : Set Values
  05 : SAVE TARGET
  06 : ELSE
  07 : NEW OBJECT: Ledger
  08 : CALL : Set Values
  09 : SAVE TARGET
  10 : END IF
  11 : END WALK
  12 : MSGBOX : "Status" : "Process completed successfully!!"
  13 : RETURN

[Function : Set Values]

  01 : SET VALUE   : Name   : $LedName
  02 : SET VALUE   : Parent: $LedParent
  03 : SET VALUE   : Opening Balance : $LedOpBal

[System : Formula]

  Ledger Exists    : $$FilterCount:Ledger:IsMyLedger
  IsMyLedger       : $Name = $$ReqObject:$LedName
```

Example 2 : Import the Ledger Master from MS Excel

3.2.2 TDL Collection to gather data from MS Excel

```
[Collection: ExcelData]
    ODBC      : "Driver={Microsoft Excel Driver
                (*.xls)};Dbq=C:\Masters.xls;DriverId=790"
    SQL       : "Select * From [Sheet1$]"
```

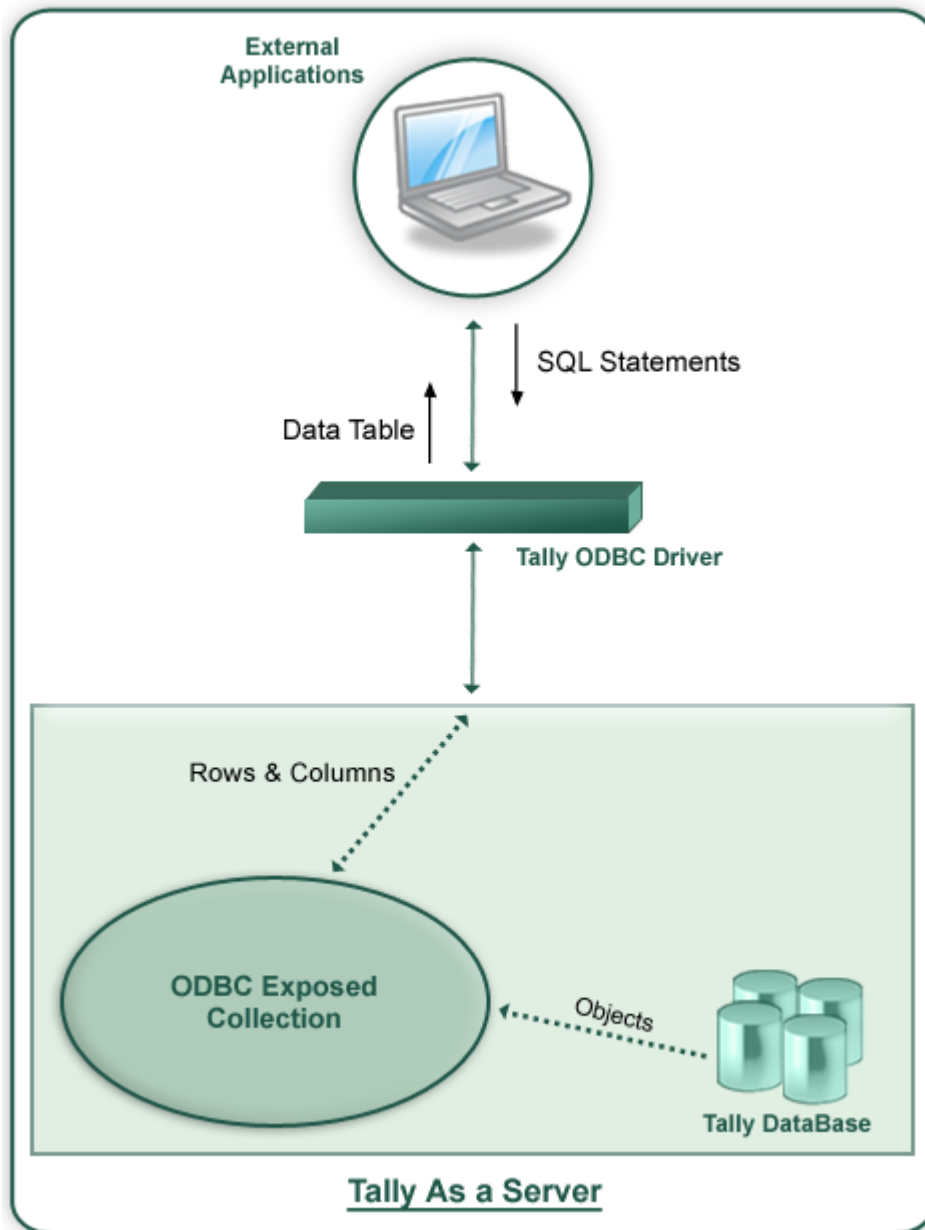
3.3 Tally.ERP 9 as a Server – Retrieving Data from Tally DB using an External Application

A Client application can access Tally.ERP 9 data in two forms

- Tables
- Calling a Procedure

3.3.1 Retrieving Data Using Tables

Tally.ERP 9 stores the data in terms of Objects. But for the external application each Object is mapped to a row and a Collection to a Table.



When Tally.ERP 9 is running as a Server to an ODBC client, not all the data i.e. Collection(s) are available to Client application. A Collection can be made available to ODBC by following two steps procedure.

1. By exposing methods of the Object(s) of the Collection.
2. By using Collection attribute 'Is ODBC Table'

Exposing Methods to ODBC

By prefixing '_' to external method(s) of an internal object or method(s) of an external Object can be exposed to ODBC.

Note:

- By default all the methods of the internal objects are exposed to ODBC
- Only First level methods of an Object can be exposed directly

Example :

```
[#Object : Ledger]
    _Difference : $ClosingBalance -$OpeningBalance
```

The code snippet alters an internal object, Ledger, to add an external method, _Difference and exposes it to ODBC.

Exposing Collections to ODBC

A Collection is exposed to ODBC by using the attribute, IsODBCTable.

Example :

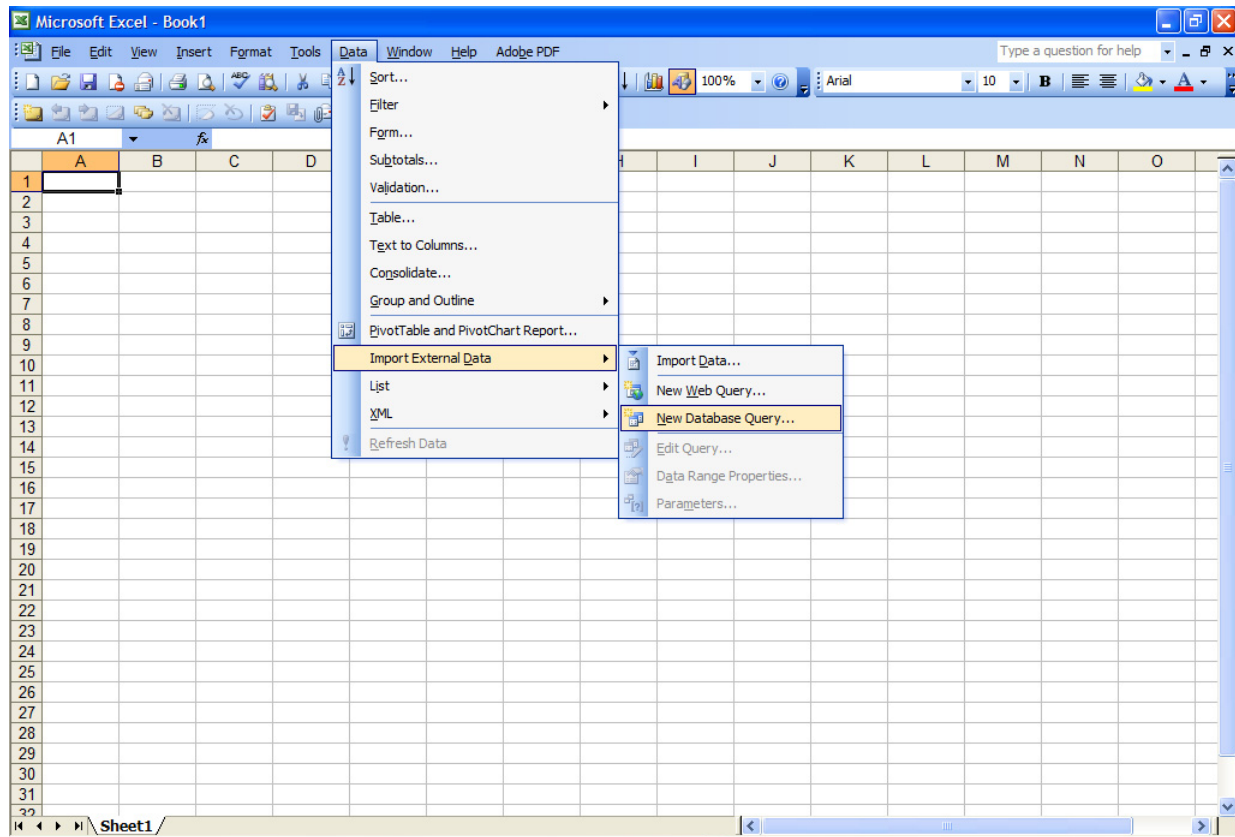
```
[Collection : Vouchers]
    Type          : Voucher
    Is ODBCTable: Yes
```

The Collection Vouchers is exposed to ODBC by using the attribute IsODBCTable.

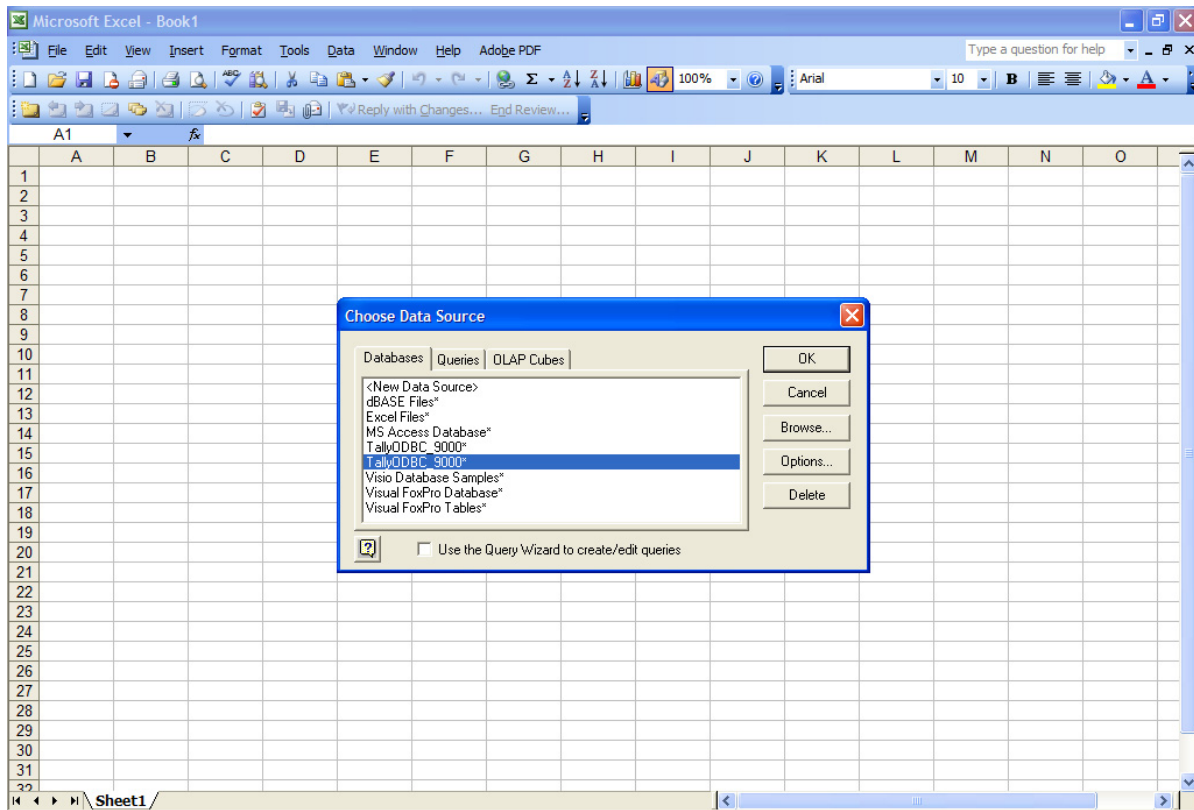
Example 1: Firing SQL statements from MS Excel to Tally.ERP 9 ODBC Server

Step 1 :- Open a New Work Book in MS Excel

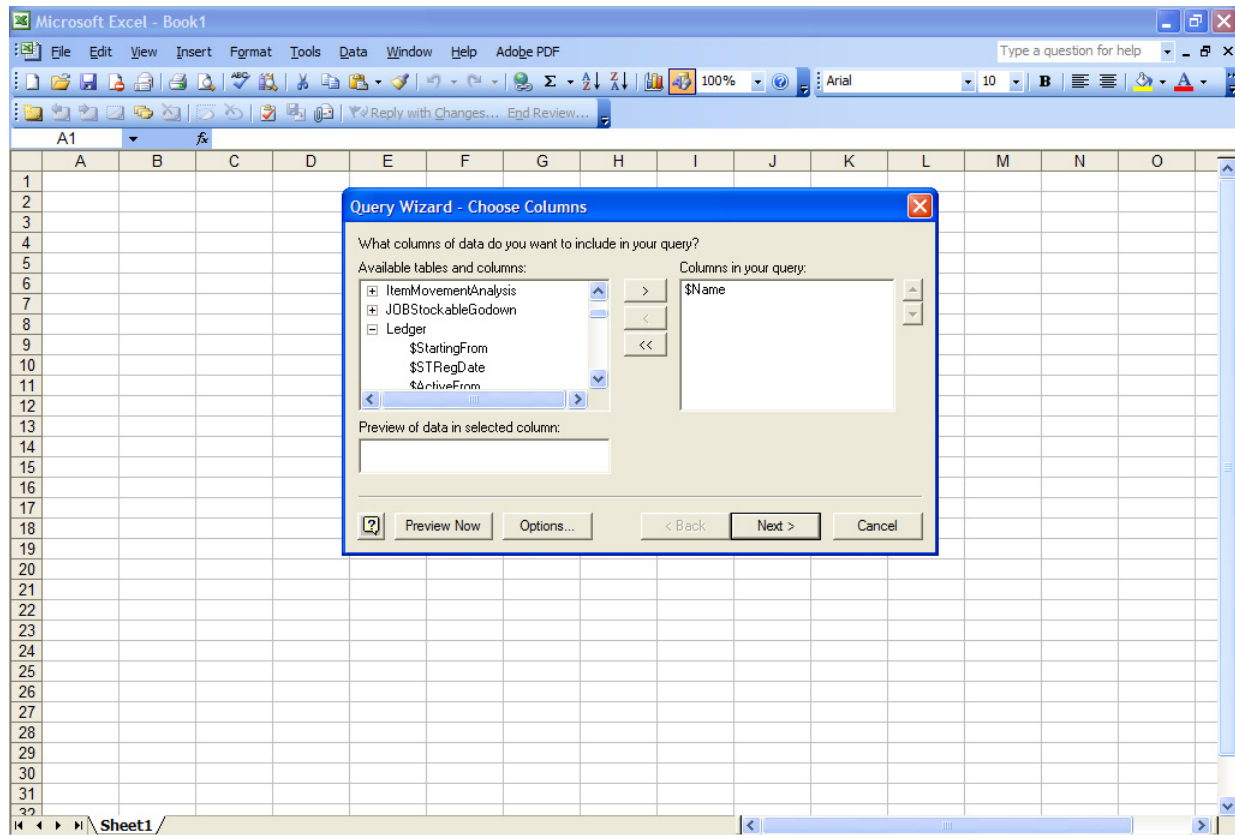
Step 2 :- Go to Data -> Import External Data -> New Database Query



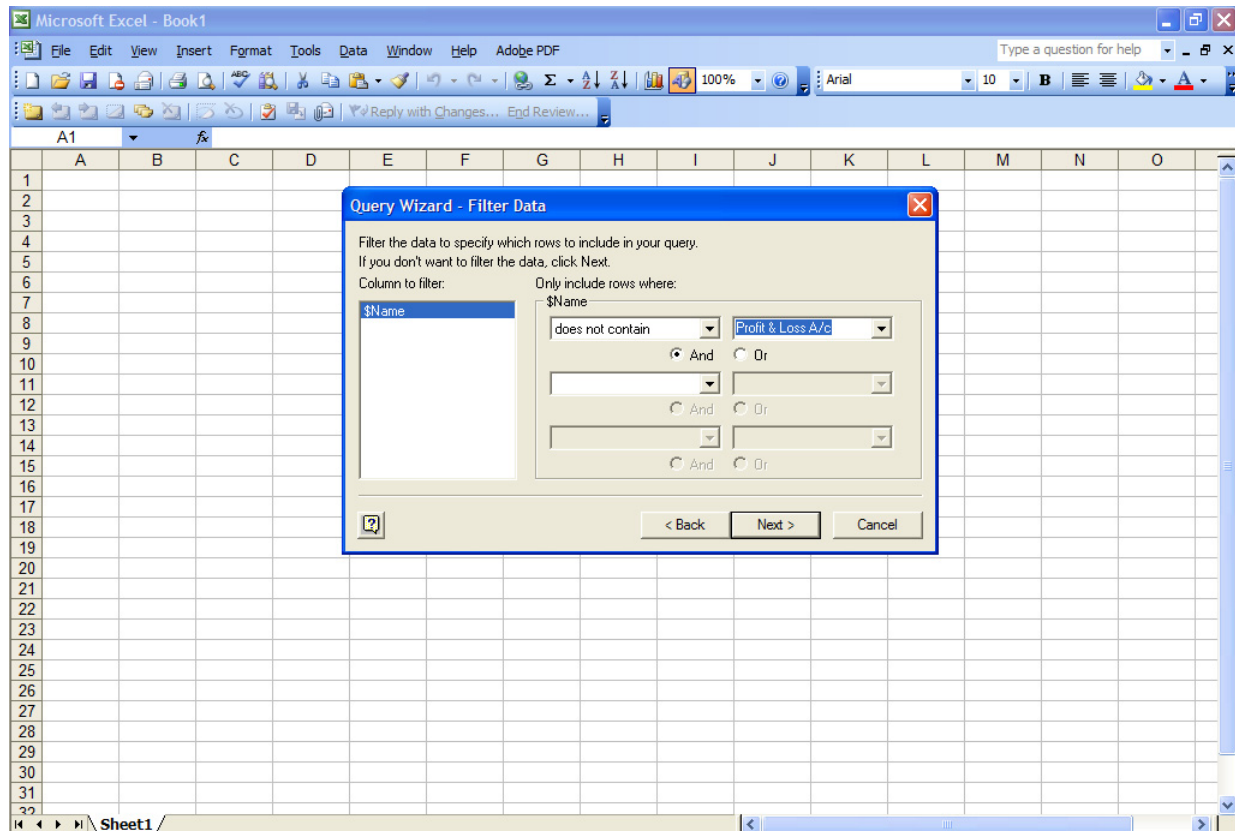
Step 3:- Select Tally ODBC Driver from 'Choose Data Source' window.



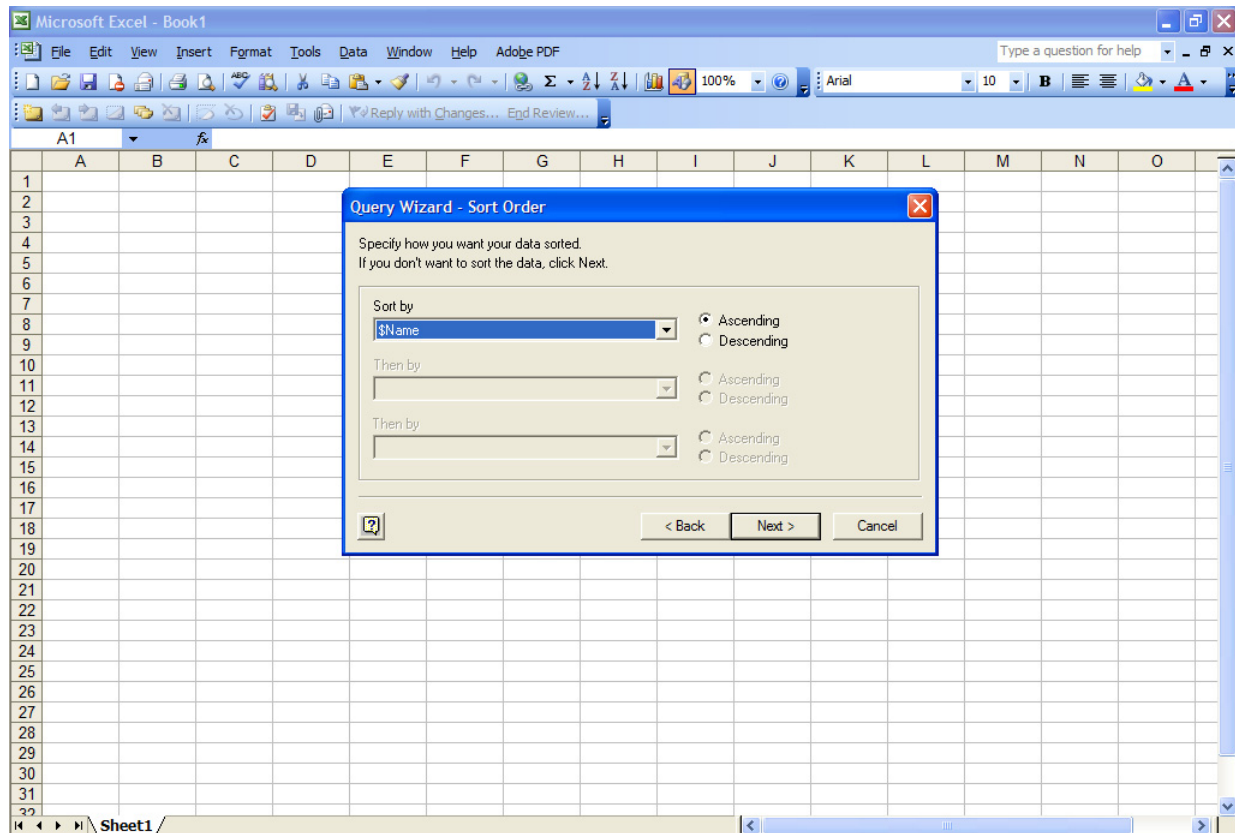
Step 4:- Select required Columns from Ledger Table from the 'Query Wizard - Choose Columns' window



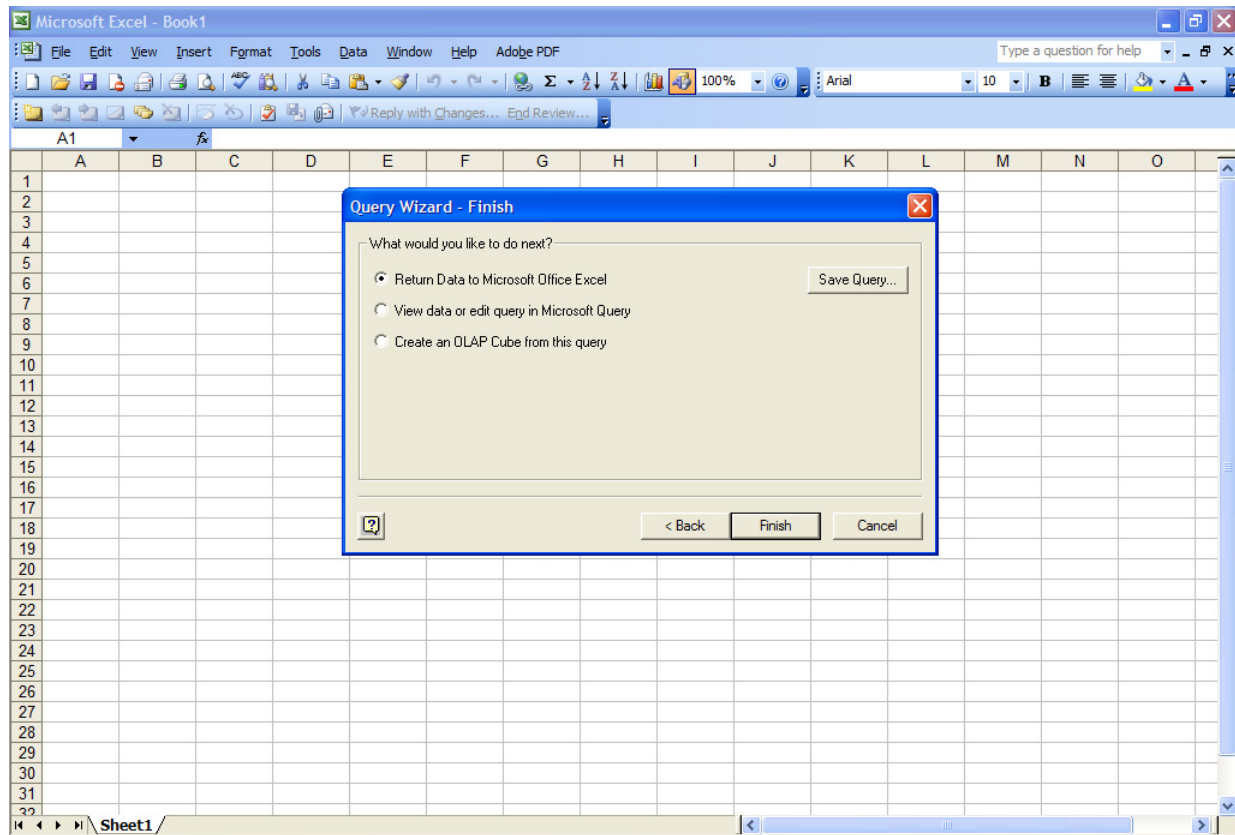
Step 5:- Filter the Data to specify which rows to include in query from 'Query Wizard - Filter Data' window



Step 6:- Sort the data from 'Query Wizard - Sort Order' window



Step 7:- From 'Query Wizard - Finish' window 'Return Data to Microsoft Office Excel'



Step 8:- View the result in Excel sheet.

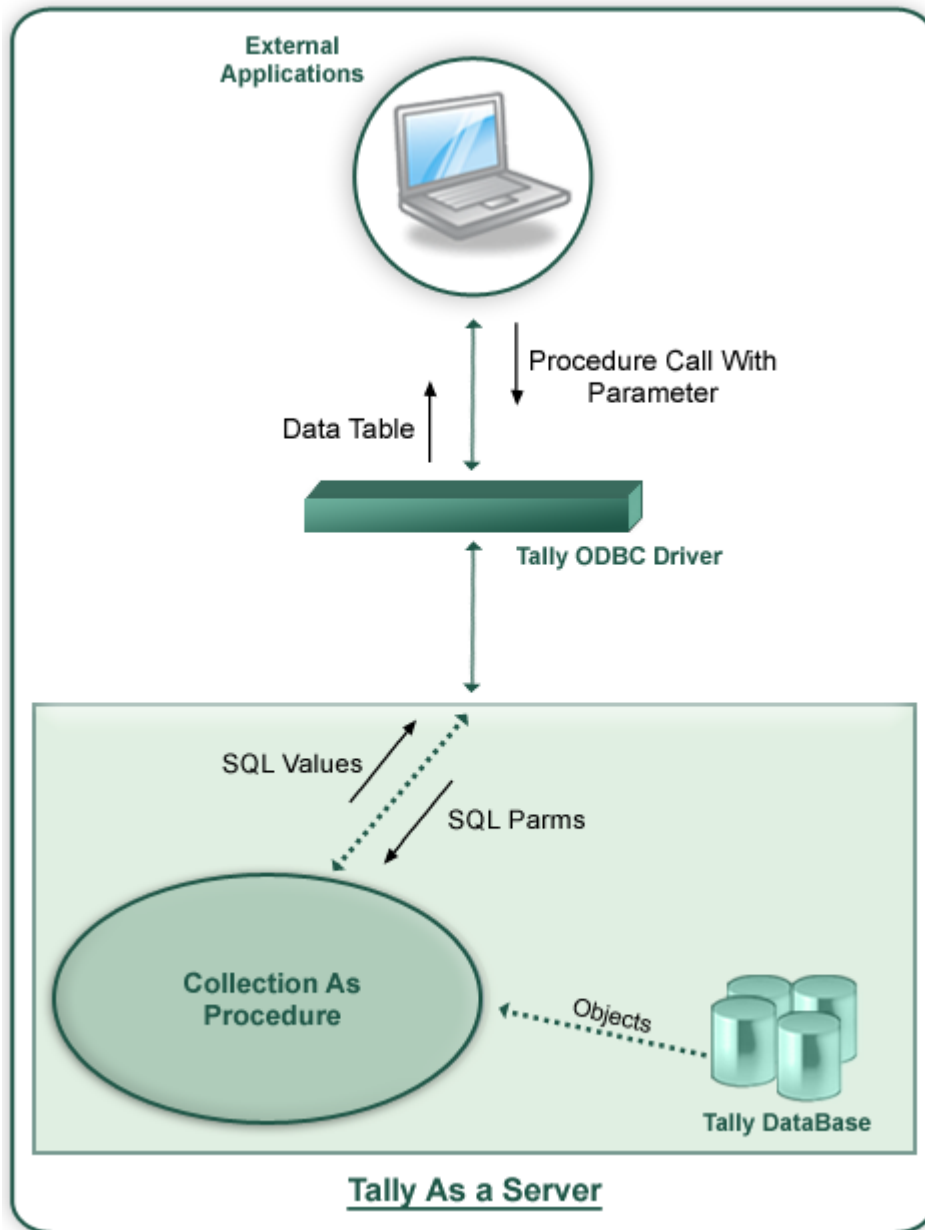
	A	B	C	D	E	F	G
1	Ledger.\$Name`						
2	Cash						
3	Purchase						
4	Sales						
5							
6							
7							
8							
9							
10							
11							
12							
13							
14							
15							
16							
17							
18							
19							
20							

Example 2: Firing SQL statements from a VB application to Tally.ERP 9 ODBC Server

Below mentioned code snippet can be used establish ODBC connection with Tally.ERP 9 and to then SQL queries can be fired.

```
Dim TallyCn As ADODB.Connection
Set TallyCn = New ADODB.Connection
TallyCn.Open "TallyODBC_9000"
Set rst = New ADODB.Recordset
rst.Open "Select $Name From Ledger", TallyCn, adOpenDynamic, adLockOptimis-
tic
```

3.3.2 Retrieving Data By Calling an SQL Procedure



A Client application can call a SQL Procedure of a Tally.ERP 9. But within Tally.ERP 9 this is a Collection with its name prefixed with an underscore. The Collection attribute, SQLParams is used to pass parameters to procedures. This Collection takes the parameter from the Client application by using the Collection attribute 'SQL Params'. The Collection attributes 'SQL Values' is used to return the values from procedure back to the client application.

3.3.3 Collection Attribute – SQLParms

This attribute is used to pass parameter(s) to a SQL Procedure. The parameter is a System variable.

SQL Parms : <Parameter>

Where **<Parameter>** is a name of the Variable

3.3.4 Collection Attribute – SQLValues

This attribute returns value to client application. SQLValues require two parameters, the Column Name and the values for the column.

SQL Values : <Column Name> : <Expression>

Where **<Column Name>** is a name of the column i.e. column header and **<Expression>** is the expression which evaluates to a value and returned back to the client application.

Usage of SQL Procedure

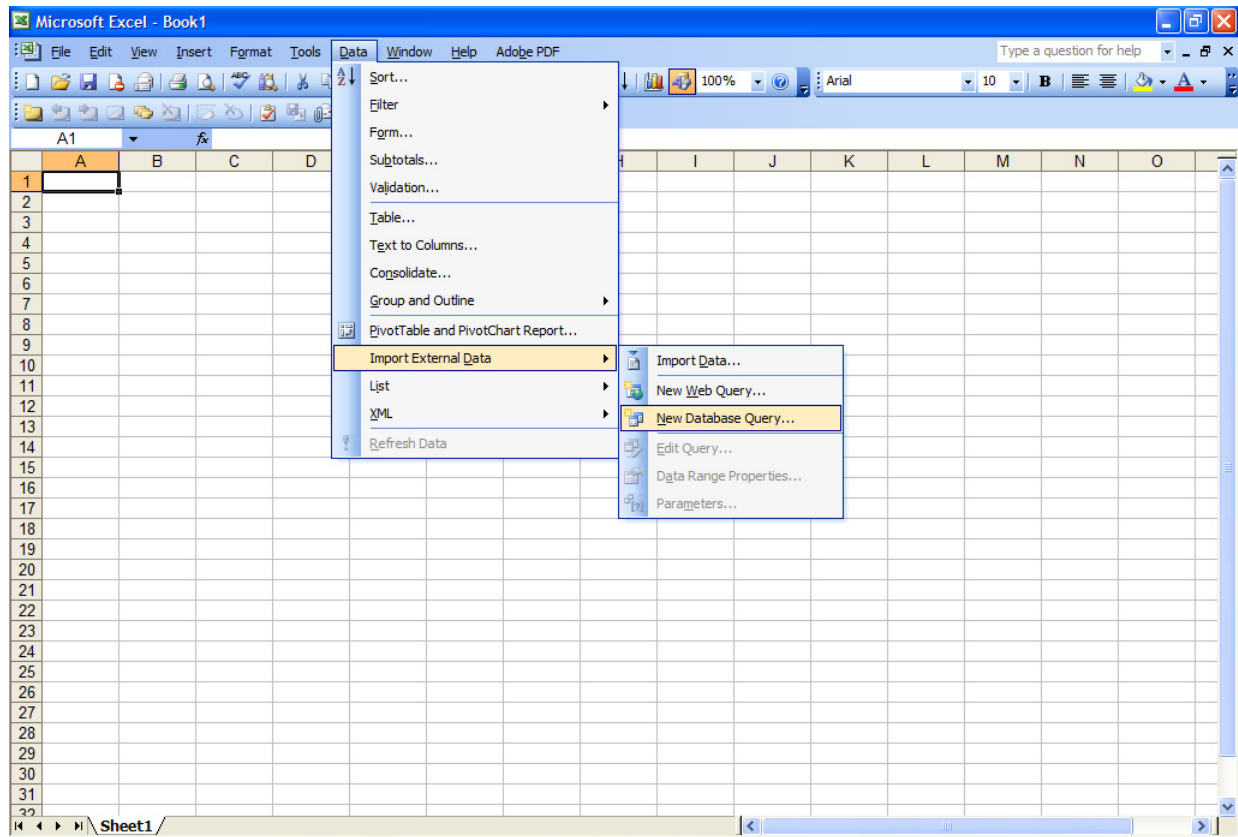
The SQL Procedure '_StkBatches' displays the Batch name and Closing Balance for a given Stock Item.

```
[Collection : _StkBatches]
    Type           : Batches
    Childof        : ##StkItemName
    SQLParms       : StkItemName
    SQLValues      : Name : $Name
    SQLValues      : Amount : $ClosingBalance
[Variable:StkItemName]
    Type           : String
```

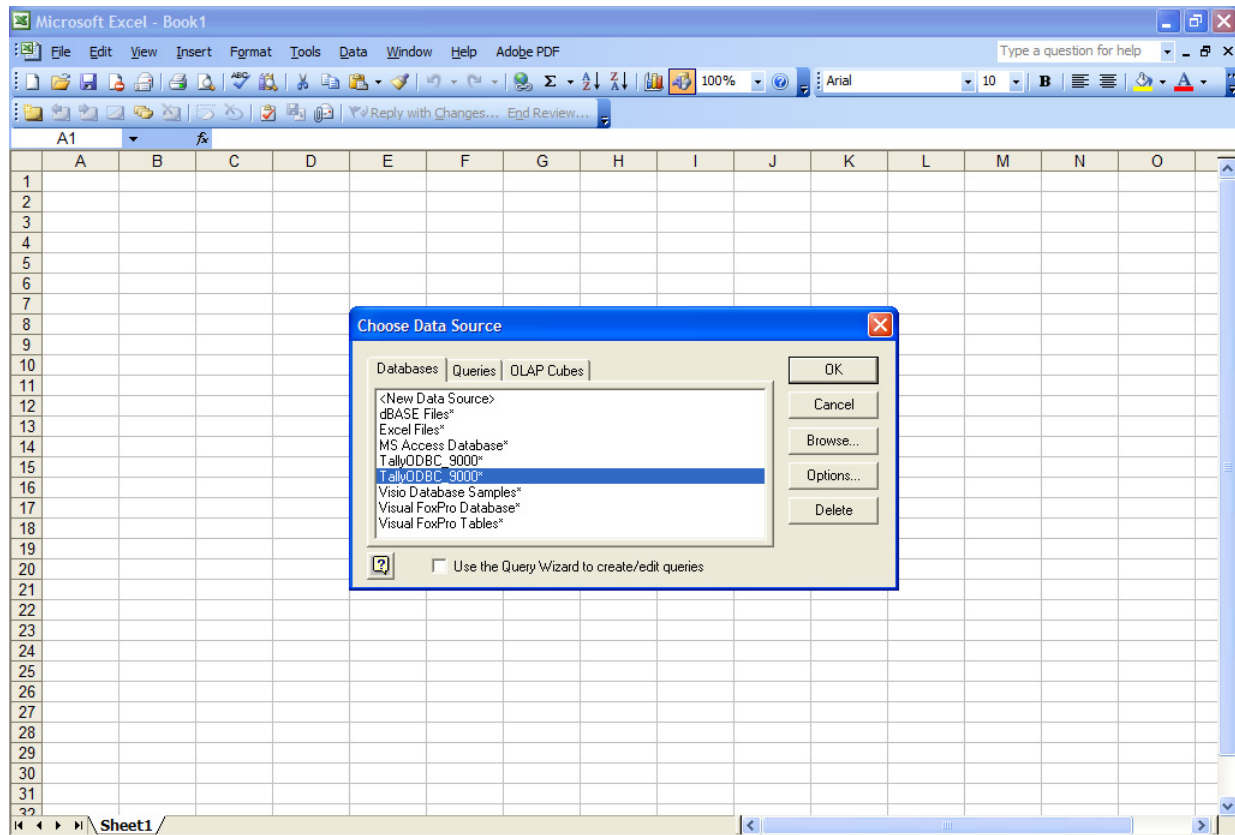
Example 1: Calling the SQL Procedure '_StkBatches' in MS Excel

Step 1 :- Open a New Work Book in MS Excel

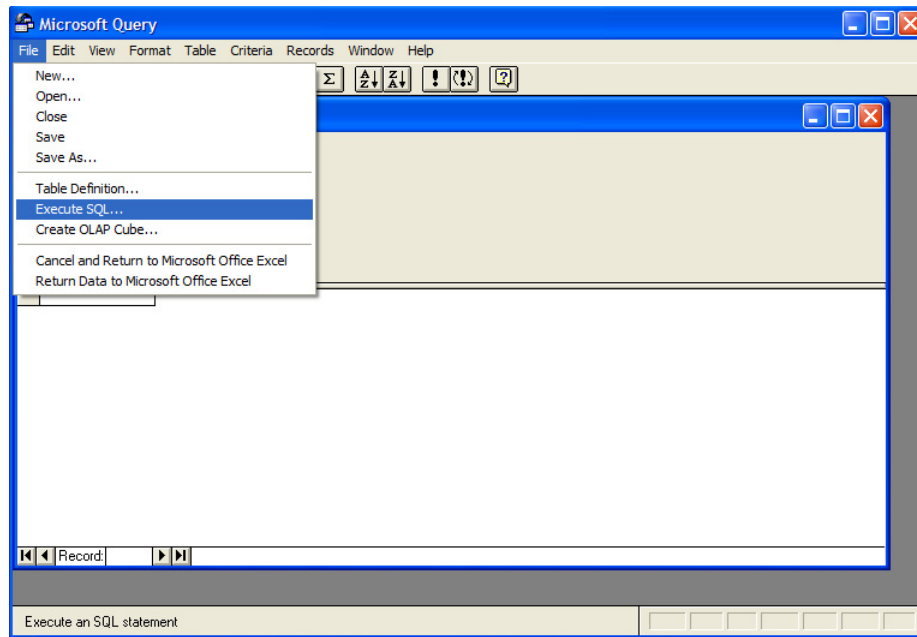
Step 2 :- Go to Data -> Import External Data -> New Database Query



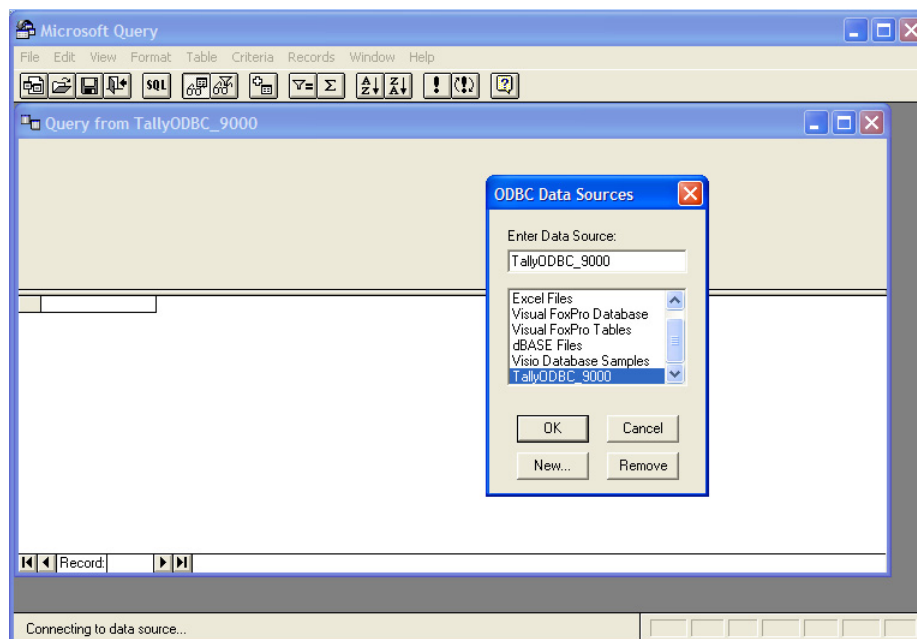
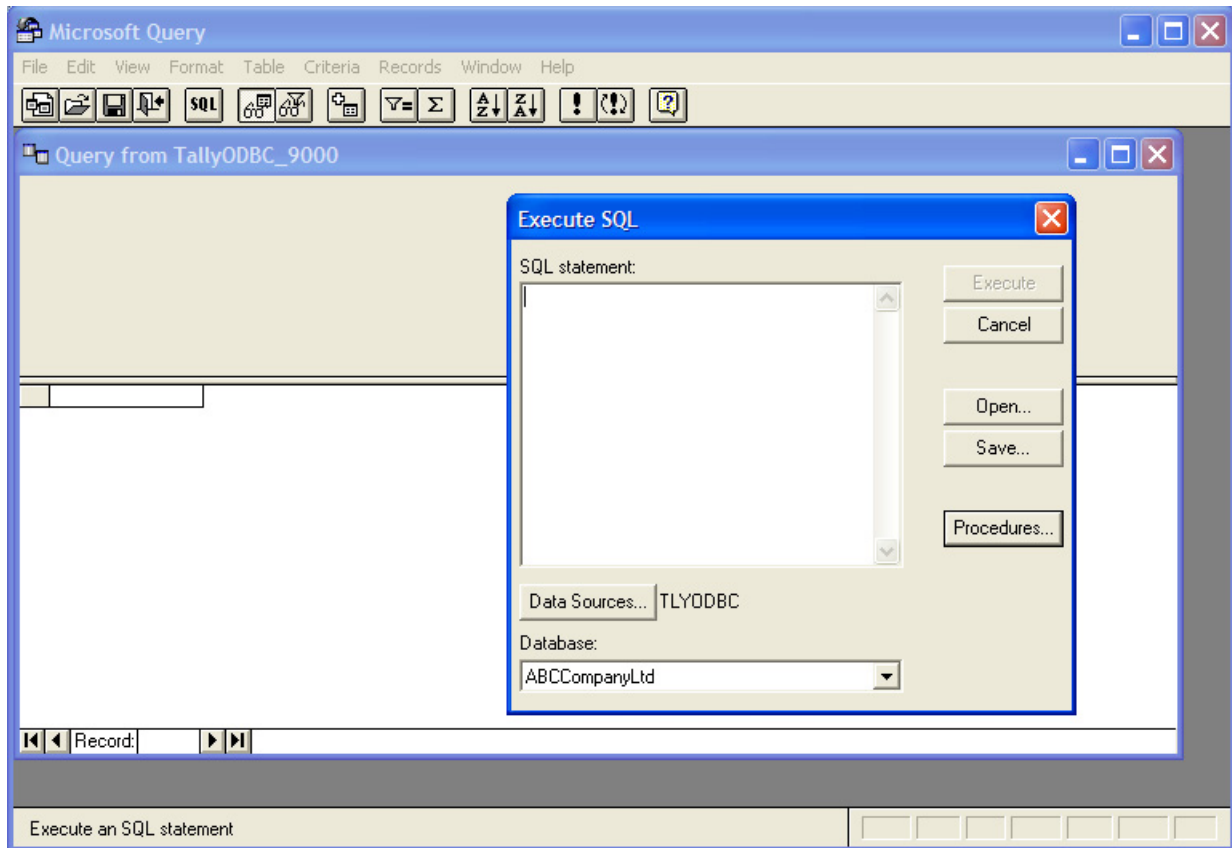
Step 3:- Select Tally ODBC Driver from 'Choose Data Source' window to open 'Microsoft Query' screen



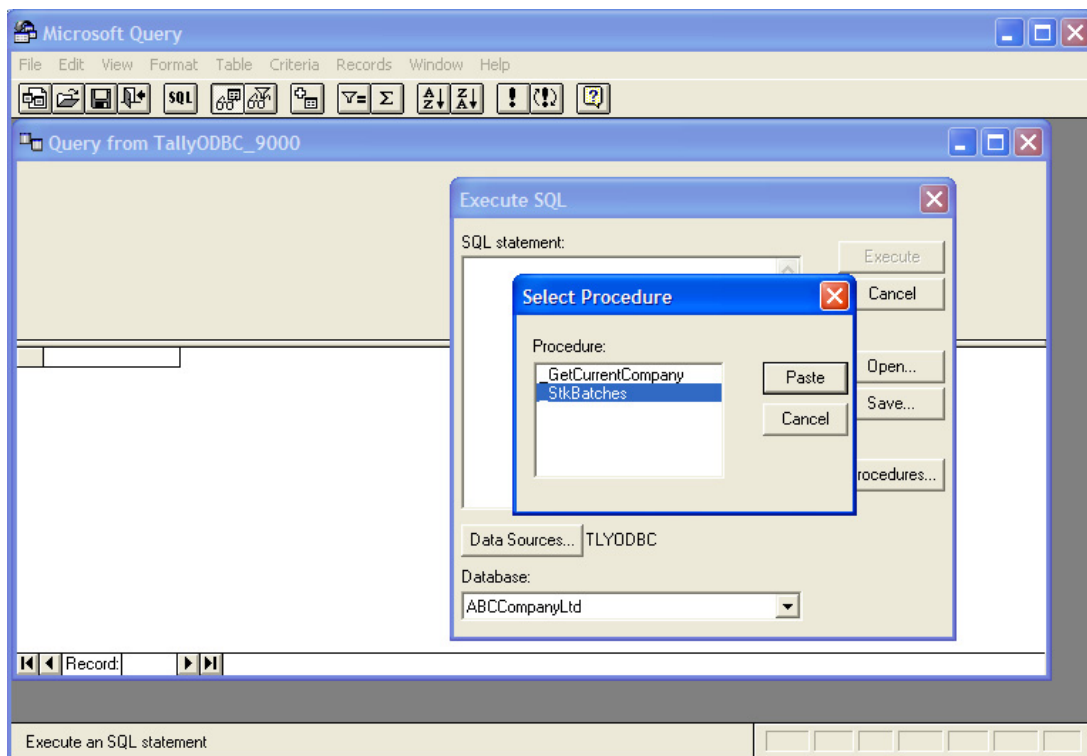
Step 4 :- Go to **File -> Execute SQL**



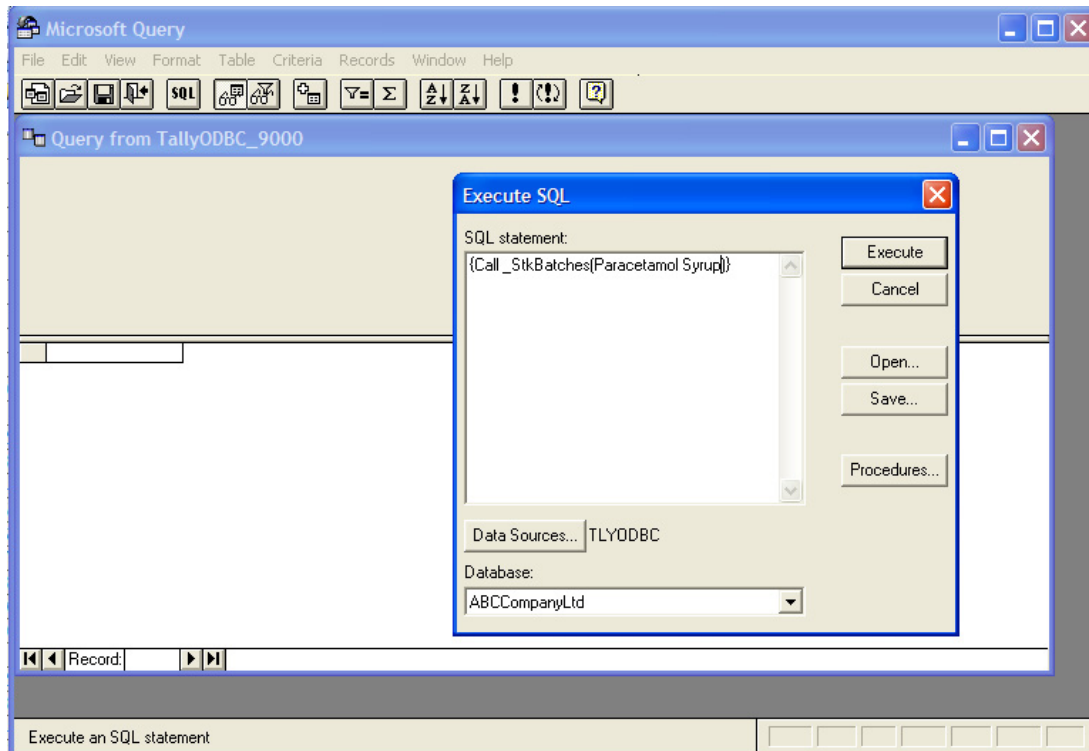
Step 5:- In '**Execute SQL**' window click on 'Procedures'



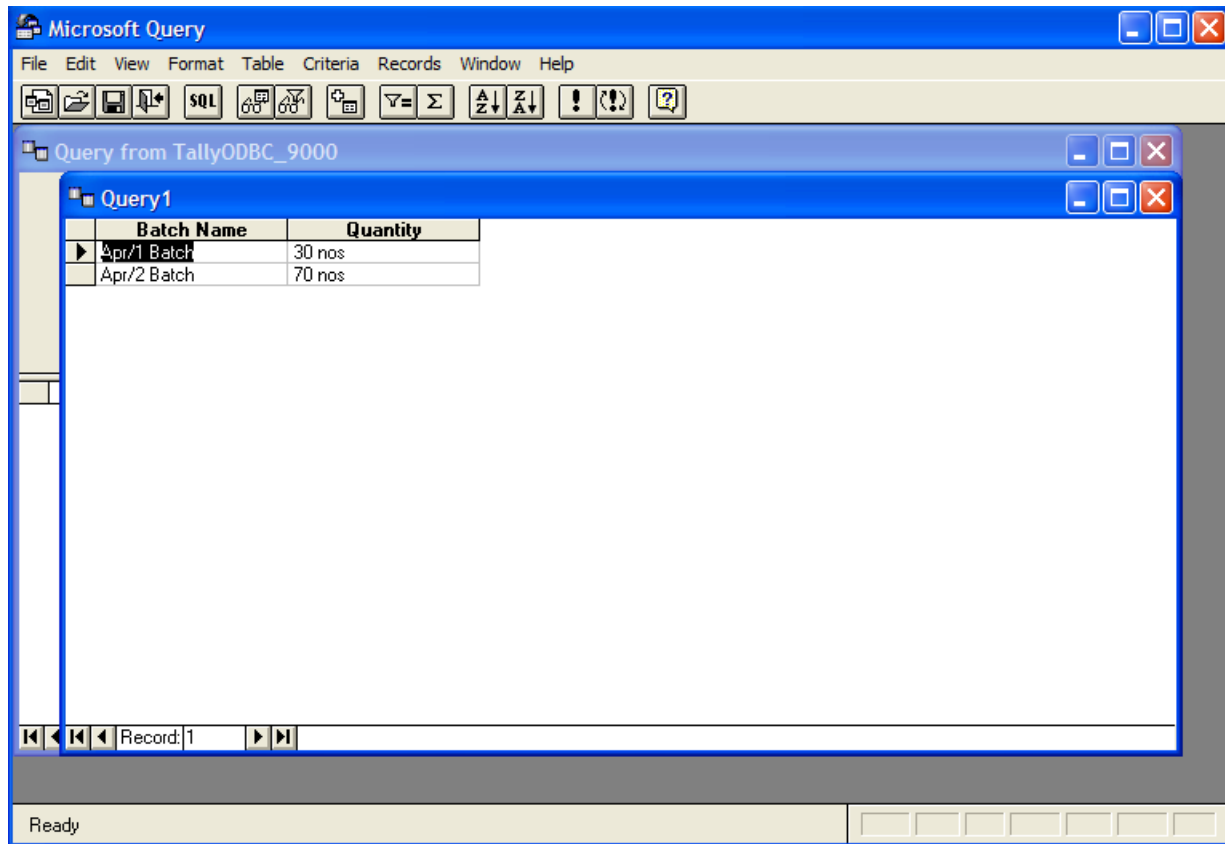
Step 6 : In 'Select Procedures' window, select procedure '_StkBatches'



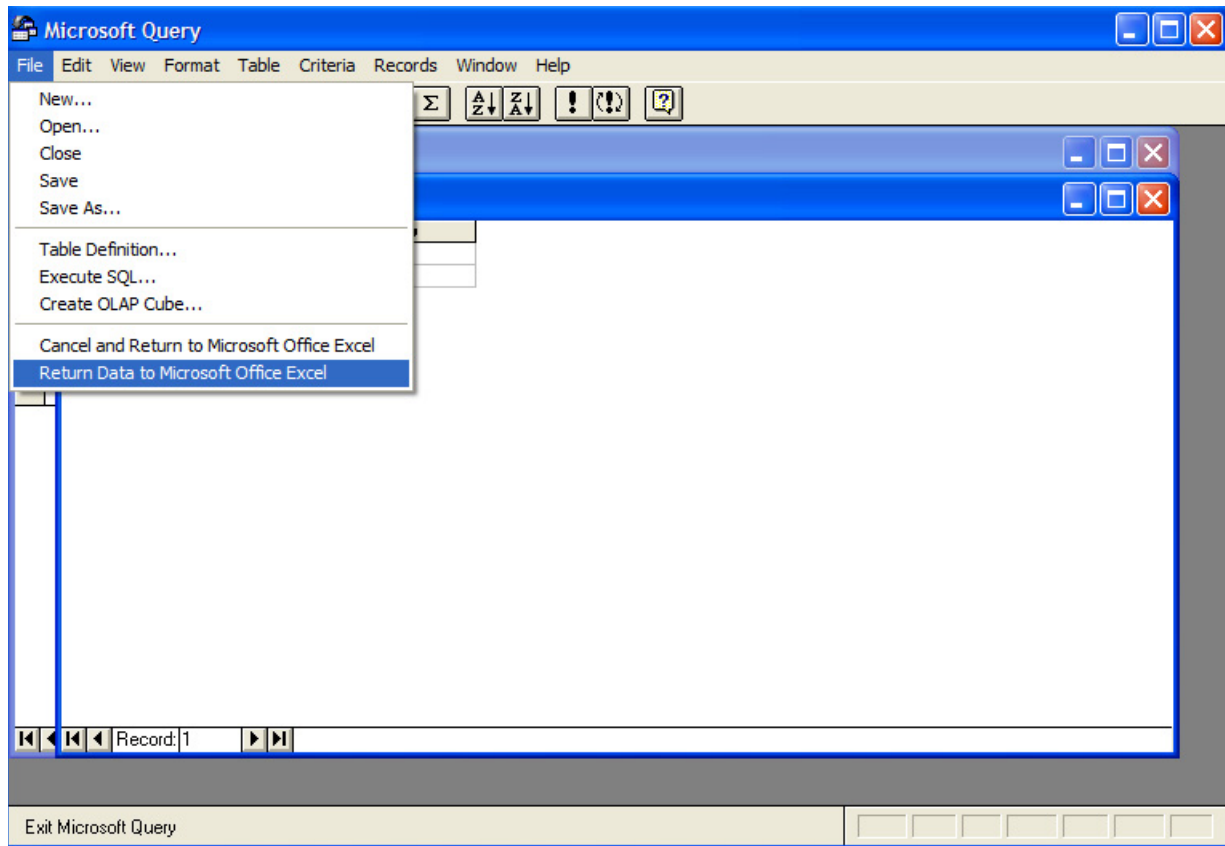
Step 7: Pass appropriate stock item name as parameter to the procedure and 'Execute'



Step 8:- View the result in 'Query1' window



Step 9:- From 'Microsoft Query' screen, Go to File -> Return Data to Microsoft Office Excel



Step 10 :- View the Result in Excel sheet .

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
1	Batch Name	Quantity													
2	Apr/1 Batch	30													
3	Apr/2 Batch	70													
4															
5															
6															
7															
8															
9															
10															
11															
12															
13															
14															
15															
16															
17															
18															
19															
20															
21															
22															
23															
24															
25															
26															
27															
28															
29															
30															
31															
32															

Example 2: Calling the SQL Procedure from a VB Application

```

Dim DBcon As New ADODB.Connection
Dim objCmd As New ADODB.Command
Dim objRs As New ADODB.Recordset
DBcon.CursorLocation = adUseClient

'Establish the connection using Tally ODBC Driver
DBcon.Open "TallyODBC_9000"
objCmd.ActiveConnection = DBcon
objCmd.CommandType = adCmdStoredProc
objCmd.CommandText = "_PartyBills"

'Pass the the Stock Item Name as Parameter
objCmd.CreateParameter (ODBCMAIN.CmbLedger.Text)

'Call the SQL procedure
Set objRs = objCmd.Execute

```

Using calculator pane for testing SQL commands

Tally.ERP 9 has an in-built SQL processor that processes SQL Select statements on collections. By default, only the collections at first level are available for selection.

Syntax

```
Select [<Method Name/s> <*>] from <Collection / Table> where <Condition>  
order by <Method Name/s>
```

Example :

- ❑ Select \$Name from Ledger
- ❑ Select \$Name, \$ClosingBalance from Ledger
- ❑ Select * from Ledger
- ❑ Select \$Name from ODBCTables
- ❑ Select \$Name, \$ClosingBalance from Ledger where \$\$IsDr:\$ClosingBalance order by \$ClosingBalance DESC
- ❑ Select \$Name, \$ClosingBalance from Ledger where \$\$IsDr:\$ClosingBalance order by \$ClosingBalance
- ❑ Select TOP 2 from Ledger