1 Introduction

The purpose of this document is to describe the Enterprise Edition of MX-Contact, which will be released at the end of December 2005.

2 Enterprise Edition

2.1 Database

The Enterprise Edition utilises a SQL Server database for data storage. No data is stored in Exchange Server Public Folders.

2.2 Migration from the Corporate Edition

For those users upgrading from the MX-Contact Workgroup or Corporate Editions (where the data resides in Exchange Server Public Folders), the Enterprise Edition includes a data migration option on the MX-Contact menu, which transfers all the existing data from the Exchange Public Folders into the SQL database.

2.3 Exchange Public Folders

A set of Outlook PST or Exchange Server Public Folders is still required. The reason the folders are still required (even though no data is stored in these folders) is that the MX-Contact custom forms, which are standard Outlook custom forms, are still utilised to display the Enterprise data and these forms are published against the corresponding Outlook folders.

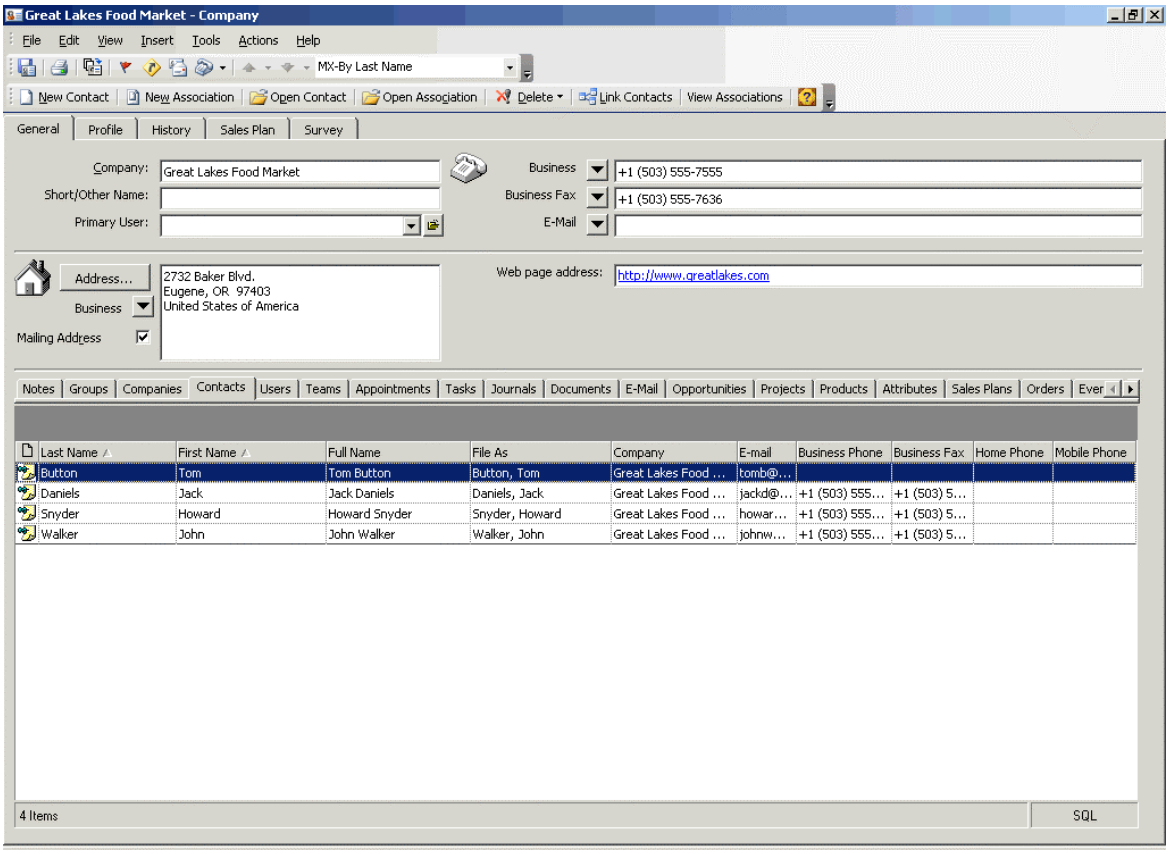
2.4 Connecting to the Database

The Folder Setup Wizard includes a dialog to establish a connection to the SQL database.

2.5 Changes to the Outlook/MX-Contact Forms

All forms remain the same as those used with the PST and Public Folder-based Editions of MX-Contact, with one exception, namely that the Outlook View Control is replaced by a custom control called MXGrid. This incorporates both the Outlook View control for the Corporate Edition and a new Janus Grid control that is used to display the SQL data.

However, because the Janus control is an Outlook-style grid, the difference between the Corporate and Enterprise Edition forms is hardly noticeable to the user, and the functionality remains virtually identical. See the Company form below by way of example:



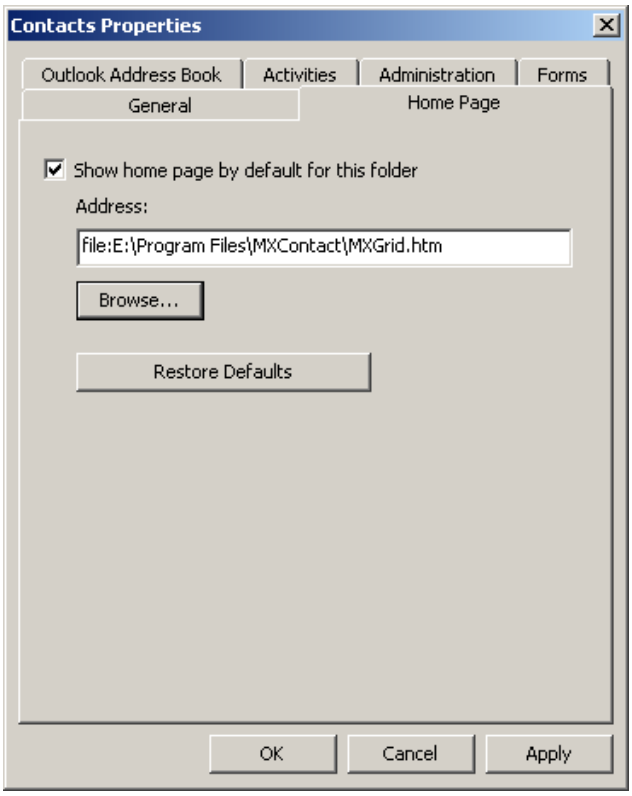
The contacts displayed in the contacts tab are being viewed **directly from the SQL database.**

2.6 Dynamic Views

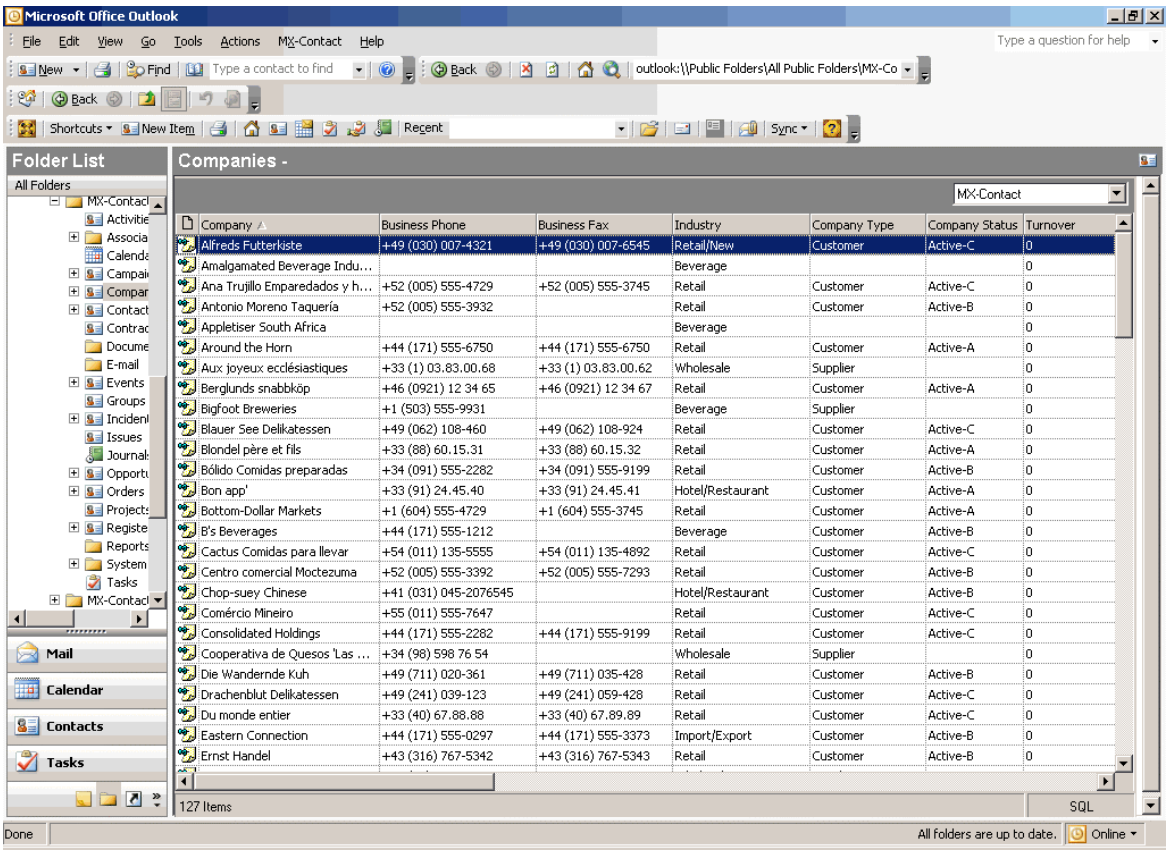
Existing Outlook Views are dynamically translated into equivalent SQL Views and these Views are then created in the SQL database. In addition one can define additional SQL views that join together different tables, which has always been a limitation of views on Exchange folders.

2.7 Folders Views

A single web page **MXGrid.htm** is set as the Home Page against each and every folder:



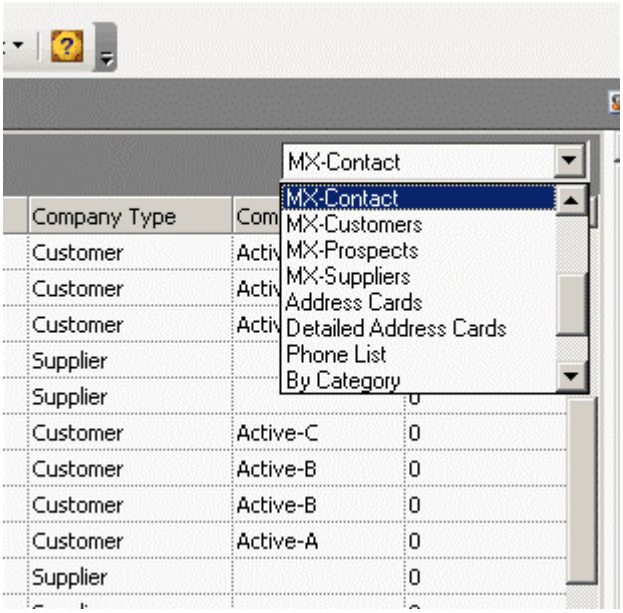
When a folder is selected the web page is displayed; this page contains a grid which displays the items in the corresponding SQL table.



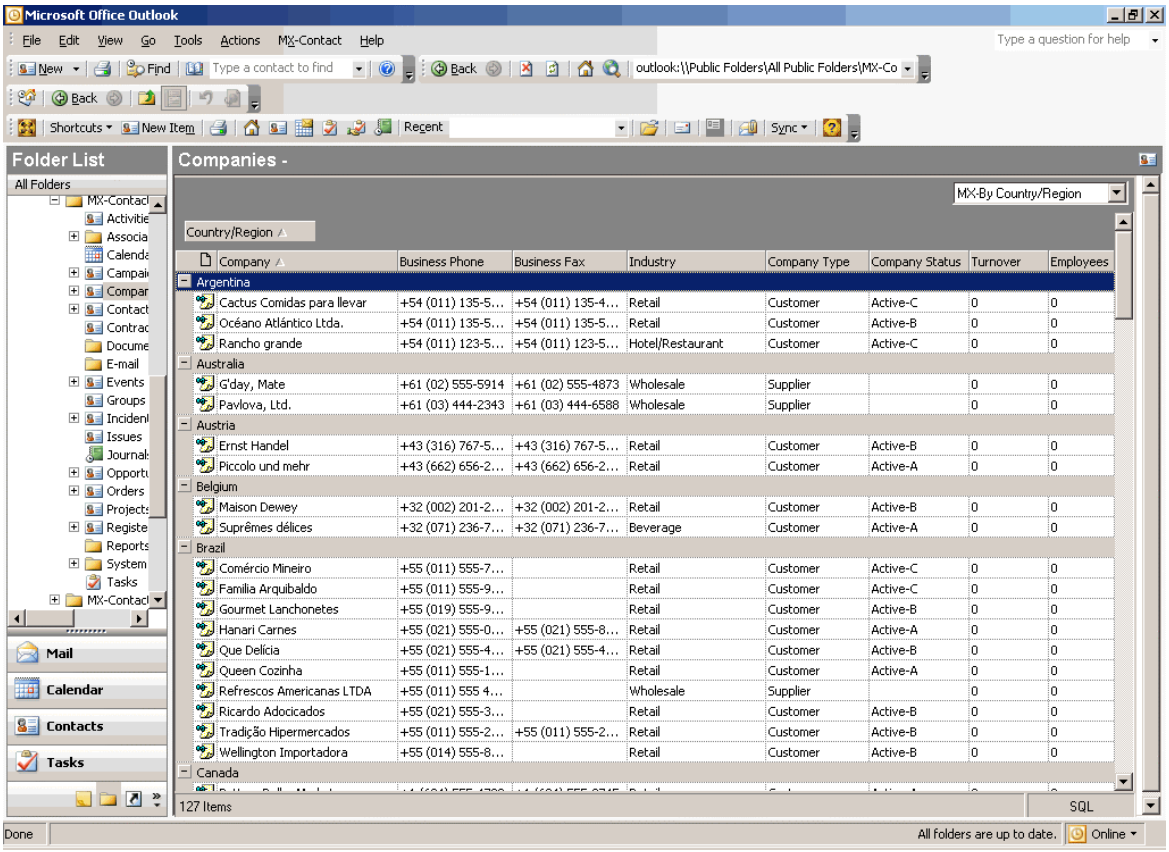
Again, the view displayed is almost identical to the standard Outlook View, even though the web page above is displaying data directly from the SQL tables.

2.8 Folder Views

All the standard Outlook Views created for that folder are displayed in the Current View dropdown:



Selecting a view displays the SQL items in almost an identical format to that seen in Outlook:



2.9 SQL Views

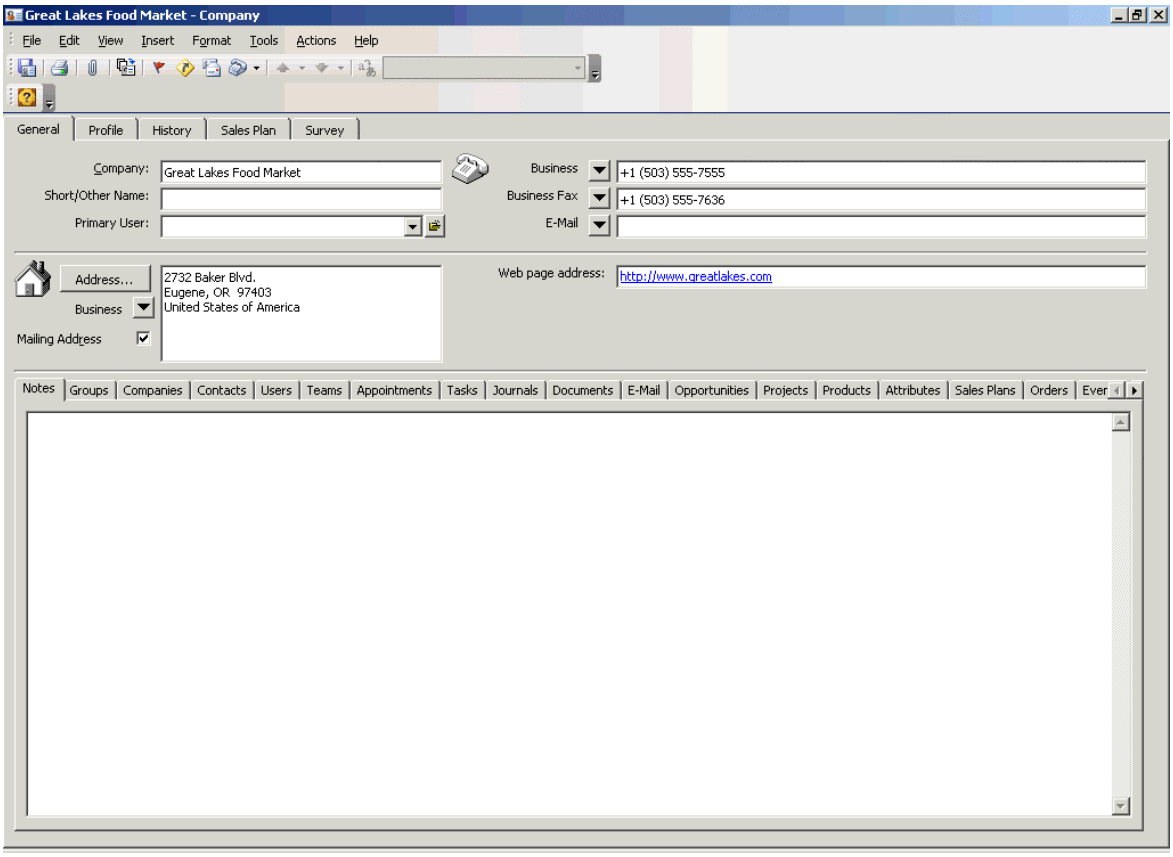
In addition to the Outlook Views, SQL Views can be created that display items in the relevant folder but joined with other tables.

So for example one can now edit the standard view showing Company to Contact Association records and by joining this table to the Contacts table, display fields from the Contacts table.

Similarly one could display a list of Contacts but show the last Meeting date by joining to the Journals table.

2.10 Opening/Editing Items

Double-clicking an item in the grid opens the item in the custom MX-Contact (Outlook) form:



When **Save and Close** is clicked, the item is saved back to the SQL table.

2.11 Linking Items

All the standard functionality in MX-Contact works the same way with the Enterprise Edition as for the Exchange-based Editions.

So for example, with a company form open, clicking on the **Contacts** tab and then clicking **Link Contacts** will open a new Explorer window that again shows the Web Page grid of all the SQL items. Selecting one or more items and clicking **Link and Close** will create associations between the company and the selected contacts.

2.12 Permissions/Security

Permissions are handled via roles in the SQL Database. In addition, one can define **Item Level Security** so that the view of companies, contacts, groups etc. is restricted to only what the user is allowed to see. This is essentially any combination of the following companies (items):

- who have that user as the Primary User (**Companies.mxUserID** = **mxUserID** of that user).
- who have that user as an associated user (in the **Associations Companies to Users** folder).
- who have a team of which that user is an associated user as an associated team (in the **Associations Companies to Teams** folder).

- in the territory that the user is allocated to.
- who match the criteria specified in a custom filter (WHERE clause) for that user. This is to cater for situations where for example a user must see all companies in the Pharmaceutical industry (mxIndustry = Pharmaceutical).

2.13 Reporting

The same set of Crystal Reports as exist for the Corporate Edition will exist for Enterprise. MX-Contact has always had the ability to report directly off the (MX-Sync) SQL Server database. There will be some change required to each report however, because different key fields are used by the Enterprise Edition to those used by the Corporate Edition which has been synced to SQL.

In addition a set of equivalent reports is being developed using SQL Server Reporting Services.

3 Benefits/Advantages of the Enterprise Edition

3.1 Scalability

Given the nature of Exchange Server's indexing mechanisms, there is a limit of some 50,000 items in any one folder in the MX-Contact Public Folder database. With the Enterprise Edition's SQL Server database, any volume of data can be managed, providing your hardware is adequately configured.

3.2 Performance

Given that SQL Server is designed as a database, and any number of specialised indexes can be created on key fields in the SQL database, the performance of the system is vastly superior to Exchange. So the speed of searching for items and linking them is significantly improved in the Enterprise Edition.

3.3 Cross Table Joins and Views

A fundamental limitation of Outlook/Exchange is that Filters can only be set using fields from the active folder. One cannot set criteria for filtering items on one folder that exist in a related folder. Naturally this can easily be achieved in a SQL Server database using SQL Views that join related tables.

3.4 Easier Integration

The fact that one's data resides in SQL means it's easier to integrate this data with other applications such as accounting/ERP systems, which are likely to have their data stored in a SQL (or similar) database.

Working with a SQL database also means that one has greater flexibility in terms of Report Writers, Mailing applications, Business Intelligence Systems, etc. that are designed to work with SQL.

3.5 Item Level Security

The fact that Exchange's security model is folder level, rather than items level, has always meant that that users can individually create views in Outlook that allow him or her full access to all the items in a folder (assuming that user has rights to view the folder).

With the Enterprise Edition, SQL Views can restrict what the user is able to see, and the administrator can restrict users from creating/editing Views, thereby limiting access to the full database.

3.6 Referential Integrity

With SQL Server tables one can create any number of primary keys and constraints to ensure that the referential integrity of the database is maintained at the database level, not by the client application. This has always been an issue with Exchange Server, which has necessitated the running of utilities like MX-Verify, to "verify" the integrity of the Public Folder database. With EE, the need for MXVerify.exe is eliminated completely.

3.7 Use of Triggers for "Rippling"

Generally rippling of fields from one table to another is not required in the EE because one can create cross-table views that contain required fields from multiple tables.

However, where rippling of data from one table to another is required, this can be achieved via triggers on the table(s), which is a more reliable mechanism than having this logic reside in the application.