Sync2DB version 1.2 Developer Guide (Brief)

Table of Contents

1. Introduction and Terminology

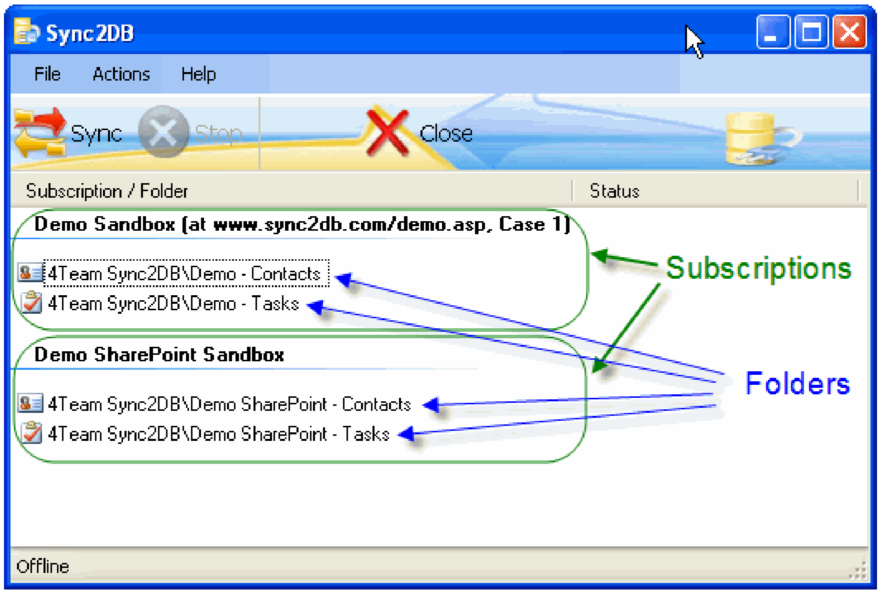
Sync2DB is a software product that synchronizes data between Microsoft® ***Outlook*** and a supported SQL-based ***database***[1].

Sync2DB consists of two main applications:

1) The Sync2DB ***Synchronizer*** application installed on each individual user's computer and used to perform the actual data synchronization.
2) The Sync2DB ***Administration Tools*** application installed on the administrator's computer and used to perform administrative actions such as generating credentials files and/or obtaining the server license. It also includes helper scripts.

For synchronization to work, special ***configuration files***[2] must be created to store connection settings and field mappings between the database tables/columns and Outlook's items/fields. This document describes how to prepare these configuration files and changes necessary on database objects in order for synchronization to work.

Groups of synchronized ***Folders*** are called ***Subscriptions***:



Demo Sandbox (at www.sync2db.com/demo.asp, Case 1)
4Team Sync2DB\Demo - Contacts
4Team Sync2DB\Demo - Tasks
Demo SharePoint Sandbox
4Team Sync2DB\Demo SharePoint - Contacts
4Team Sync2DB\Demo SharePoint - Tasks

Figure 1: Subscriptions and Folders in the main Synchronizer window

[1] Only Microsoft® SQL Server was supported in version 1.0. MySQL® is also supported starting from the version 1.2. Future releases are planned to allow data synchronization with Oracle® RDBMS and other SQL-based databases. All samples, scripts and instructions contained within this document are specific to Microsoft® SQL Server unless otherwise stated.

[2] In version 1.0, the configuration files were stored on each user's computer only. The version 1.2 and later allows these to be stored centrally - in the database as well as in the file system of a server (see topic "3.10 Remote Subscriptions").

2. Preparing database tables to be synchronized

Synchronization imposes several requirements for each database table to be synchronized:

- A primary key must be properly defined
- Special synchronization columns must be defined
- Special synchronization support table and triggers must be defined

2.1 Steps to prepare database tables for synchronization

These are the basic steps needed to prepare database tables that are to be synchronized:

1. **Ensure that a primary key is defined.** Refer to topic "2.2 Primary key requirements" for more details.
2. **Ensure that new keys are generated as expected.** If the primary key consists of a single column and this column is described as IDENTITY (AUTO_INCREMENT), no further action is required regarding the primary key. Otherwise, the **stored procedure for primary key generation** should be created and used**.** E.g.: refer to the SQL script file S*harePoint_CreateSync2DBObjects.sql* in directory *(Administration Tool Application folder)\Documentation\Samples\MSSharePoint\MSSQL Scripts* or Custom_CreateObjects.sql in directory *(Administration Tool Application folder)\SQL Scripts\MySQL\* (stored procedure *spFT_GetNextId*)
3. **Adjust and execute three (3) SQL scripts** that are included in the Administration Tools installation package. Refer to topic "2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers)" for more detailed instructions.
4. **Establish an SQL user that will be used by Sync2DB** to connect to the database. Create it if necessary.
5. **Assign one of the database roles** (*Sync2DBReader* or *Sync2DBWriter*) to the SQL user established in step 4. If access needs to be read-only, assign *Sync2DBReader*; otherwise, assign *Sync2DBWriter*. (These database roles were created via one of the scripts in step 3).

2.2 Primary key requirements

Sync2DB requires that every data table to be synchronized have a primary key which is a column or combination of columns that contain values that uniquely identify each record in the table. Sync2DB supports primary keys that consist of a single column or a compound key made up of several columns (also known as a composite primary key). Sync2DB also requires that primary keys are **not reused**. That is, if a record is deleted, the primary key of the deleted record is never used for another record.

2.3 Metadata requirements

Sync2DB requires that every data table to be synchronized have metadata defined as follows:

- ***Change tracking columns.*** Five (5) special metadata columns are needed:

- 1) The id of the Sync2DB user who inserted the record. The default column name is **ft_InsertId**. Sync2DB users can be the same as users of a corporate system whose database participates in synchronization. It can have one of the data types that are described within the file *Common_CreateObjects.sql* .
- 2) Anchor that indicates when the record was inserted. The default column name is **ft_InsertAnchor**. It can have one of the data types that are described within the file Common_CreateObjects.sql. It must take on a new value for every inserted record.
- 3) The id of the Sync2DB user who last updated the record. The default column name is **ft_UpdateId**. The data type must be the same as that of **ft_InsertId**.
- 4) Anchor that shows when the record was last updated. The default column name is **ft_UpdateAnchor**. The data type must be the same as that of **ft_InsertAnchor**. It must take on a new value every time a record is changed.
- 5) The id of the Sync2DB client (i.e. combination of a Windows User and a computer) who last inserted or updated the record. The default name is **ft_ClientID**. It must have the data type *uniqueidentifier* (for Microsoft® SQL Server), and one of the data types that are described within the file *(Administration Tool Application folder)\SQL Scripts\MySQL\Common_CreateObjects.sql* (for MySQL). ***Note:*** *Sync2DB user and Sync2DB client are not the same!*

We recommend using the stored procedure *spFT_AddMetadataColumns* (see topic "2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers). Specify the column names using the appropriate parameters (user-defined variables for MySQL®): @InsertIdColumn, @InsertTimestampColumn, @UpdateIdColumn, @UpdateTimestampColumn, @ClientIdColumn. This stored procedure creates the specified columns, if they don't exist, or uses the already existing ones. Note that you should provide the already existing columns only if they fit the requirements above.

- ***Tombstone table*** (optional). If deletions are to be synchronized, a separate table with a special structure to track deletion operations is needed. As a rule, the tombstone table should have a primary key column or columns (the same ones, as the data table has) and three (3) special metadata columns or ***deletion tracking columns***.

  The Tombstone table requirements are:
  - The name of tombstone table must be formed from the name of the data table by adding the suffix "*_Tombstone*".
  - Every record deleted from the data table must insert a record into the tombstone table.
  - The tombstone table must contain the same primary key columns as the associated data table.
  - The deletion tracking column requirements are:
    - The id of the Sync2DB user who deleted the record. The default name is **ft_DeleteId**. The data type must be the same as that of **ft_InsertId** and **ft_UpdateId**.
    - Anchor that shows when the record was deleted. The default name is **ft_DeleteAnchor**. The data type must be the same as that of

ft_InsertAnchor and **ft_UpdateAnchor**. It must take on a new value every time a record is deleted.
  - The id of the Sync2DB client (i.e. combination of a Windows User and a computer) who deleted the record. The name and data type must be the same as that of the appropriate column in the data table (**ft_ClientID** by default).

  The previously mentioned stored procedure *spFT_AddMetadataColumns* will automatically create a tombstone table unless the parameter *@CreateTombstoneTable* is set to 0. (*@CreateTombstoneTable=0* means "Do not create tombstone table.").

- **Values of the three anchor columns** (**ft_InsertAnchor**, **ft_UpdateAnchor** and **ft_DeleteAnchor**) **must form an ascending sequence** (i.e. every new value for any of these columns must be greater than any previous value of any of these columns).

  Unless the data table already has columns meeting the Sync2DB requirements and the native application properly handles them (i.e. Microsoft® SharePoint), special **tracking triggers** must be defined on each data table to properly fill in the change tracking columns and tombstone table. The previously mentioned stored procedure *spFT_AddMetadataColumns* will automatically create tracking triggers unless the parameters @CreateTriggerOnInsertUpdate, @CreateTriggerOnDelete, and/or @CreateTriggerOnInsertTsT are set to 0. (A value of "0"means "Do not create the appropriate trigger").

2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers)

The following simple procedure is recommended to create the required change tracking columns:

1) Execute script
   (Administration Tool Application folder)\ SQLScripts\MSSQL[3]\Common_CreateObjects.sql
   for each database containing tables to be synchronized. This script creates the stored procedure *spFT_AddMetadataColumns*, which allows the change tracking columns, tombstone table, and/or tracking triggers for several typical variants to be created easily. To specify these variants, use the stored procedure parameters (for Microsoft® SQL Server) or user-defined variables (for MySQL®). See comments in the stored procedure definition for parameters (for Microsoft® SQL Server) and comments within the file *Custom_ForEachTable.sql* (for MySQL®). The detailed instructions on how to execute a particular script can be found at the top of the script file itself.
2) Make your own copy of the script file
   (Administration Tool Application folder)\SQL Scripts\MSSQL\Custom_CreateObjects.sql
   and adjust it according to the specifics of your database.
   This script:
   a. **Creates the database roles Sync2DBReader and Sync2DBWriter[4]** needed to provide easy management of permissions for data tables. Scripts *Custom_CreateObjects.sql* and *Custom_ForEachTable.sql* (described below) set all of the required permissions for these roles. The only thing left to do is to assign one

[3] The folder "MSSQL" contains scripts for the Microsoft® SQL Server. Other folders ("MySQL", "Oracle", etc.) contain scripts for relevant DBMS servers.
[4] Only script for the Microsoft® SQL Server.

of these roles to the SQL user that will be used by Sync2DB to connect to the database. If access needs to be read-only, then assign *Sync2DBReader*; otherwise assign *Sync2DBWriter*. All this is optional as permissions can be customized as desired. In this case just comment out these lines in the script.

   b. **Creates the sample ftUsers table** needed to store Sync2DB users and provide authentication.
      If authentication at the Sync2DB level is not required, that is, everyone can have access to your database using Sync2DB or authentication is provided at the Microsoft® SQL Server level only, the creation of this table can be bypassed by commenting out these lines in the script. Two options are available to utilize the user information already contained in the systems such as CRM, CMS, Microsoft® Sharepoint, Microsoft® Project Server, etc:
      i. Use only the pre-existing tables by creating an authentication procedure similar to the procedure of your corporate system, for example, all CRM users of Sales department (CRM group "Sales") will be able to synchronize data using Sync2DB Synchronizer. In this case, bypass the creation of this table by commenting out these lines in the script.
      ii. Use both the pre-existing tables and the new table **ftUsers**. In this case, the user must exist in both tables. This option allows synchronization access to only a selected number of users.
   c. **Inserts records into the table ftUsers**. Change sample values (UserName, RealName, user passwords) to real ones. Comment out these lines if **ftUsers** isn't going to be used.
   d. **Creates the sample *spFT_AuthenticateUser* stored procedure**. If authentication is at the Sync2DB level, this stored procedure is mandatory. Change its content according to the required authentication procedure. The sample contains checking of the user name and password against the data from the **ftUsers** table.
   e. **Creates the sample *spFT_GetNextId* stored procedure**. If the custom stored procedure for primary key generation method is used, change the script according to your requirements, otherwise, comment these lines out.

3) Execute the just prepared script. You can execute all of these scripts several times to adjust some settings without reverting the changes made to your database.
4) Make your own copy of the script file
   (Administration Tool Application folder)\SQL Scripts\MSSQL\Custom_ForEachTable.sql
   for each of the data tables.
5) The following actions should be implemented with each created script file[5]:
   a. Open the script file with Microsoft® SQL Server Management Studio.
   b. Connect to the SQL server where the DataTable is located (SQL server administrator or this database owner credentials are required).
   c. In the combo-box "Available Databases", select the database where the data table is located.
   d. Use the Specify Values for Template Parameters command of Management Studio (Ctrl-Shift-M) to fill in the owner and name of the data table in all the

[5] This instruction is suitable for Microsoft® SQL Server only. For instruction for other DBMS servers please refer to relevant file Custom_ForEachTable.sql.

appropriate places in the script. Simply press Ctrl-Shift-M and enter the owner and the name of the data table in the dialog.

e. If needed, add optional parameters into the call of stored procedure *[dbo].[spFT_AddMetadataColumns]*. This stored procedure allows easy creating of the change tracking columns, tombstone table, and/or tracking triggers for several typical variants specified through the parameters. See comments for parameters in the stored procedure definition (in the file *Common_CreateObjects.sql*). To use the recommended (default) parameter values, call this stored procedure with only one parameter, the name of data table. This script also grants all of the required permissions to the database roles Sync2DBReader and Sync2DBWriter.
f. Execute the script by pressing <F5>. The stored procedure *[dbo].[spFT_AddMetadataColumns]* prints a report during the execution. **Please review it carefully, especially the red error messages**. The last line of the report must be **copied and pasted** into the `<trackingFields>` node in the subscription configuration file for the data table.

Notes:
1) The spFT_AddMetadataColumns stored procedure doesn't change the columns, tombstone table, and indexes. It only creates them if they don't already exist. If these need to be changed, delete them before repeating the execution via, for example, the Object Explorer of Microsoft® SQL Server Management Studio. Unlike columns, tombstone table, and indexes, triggers will be updated if they exist and therefore don't need to be deleted before repeating the execution.

3. Preparing configuration files

After preparing the database tables to be synchronized, the configuration files can be created.

3.1 Steps to prepare configuration files

1. Prepare the **Subscription file** as described in the "3.3 Subscription file" section.
2. Prepare the **Mapping file(s)** as described in the"3.6 Mapping files." section.
3. (Optional) If the user name (login) and password are used for direct connection to the SQL server, prepare a **Credentials file** as described in the "3.4 Credentials file" section.
4. (Optional) Prepare the **Rules/Agreement file** as described in the "3.5 Rules/Agreement file" section.
5. (Optional) If the "Corporate Site" and "Corporate Help" menu items in the "Help" menu of the Synchronizer are to be customized, prepare the **Sync2DB.exe.config file** as described in the "3.7 Sync2DB.exe.config file" section.
6. Prepare a **configuration package (".2db" file)** from previously prepared files as described in the "3.8 Making the Configuration package (".2db" file)" section.
7. **Test the prepared configuration package** before distributing to other users (open it via Synchronizer and try to synchronize every folder).
8. **Distribute the configuration package** to other users as described in the "3.9 Distributing the configuration package (".2db" file)" section.

3.2 Overview (required files and directory structures)

The Sync2DB Synchronizer searches for configuration files that are located in the "*Configuration*" folder within the Sync2DB User Application Data folder. Typically, this is *C:\Documents and Settings\(WINDOWS_USER_NAME)\Application Data\4Team\Sync2DB\* (for a Windows XP system) and *C:\Users\(WINDOWS_USER_NAME)\AppData\Roaming\4Team\Sync2DB\* (for a Windows Vista system). If this folder doesn't exist when Sync2DB Synchronizer starts, Sync2DB Synchronizer will create it along with the "*Configuration*" subfolder and copy to that subfolder all the contents of the Sync2DB Synchronizer Application folder's "*Configuration*" subfolder. Typically, this is *C:\Program Files\4Team Corporation\4Team Sync2DB*.

The "*Configuration*" subfolder under the Sync2DB User Application Data folder contains the following:

- **"*CdoConstants*"** subfolder, with the service files that **must not be changed**.
- **"*Subscriptions*"** subfolder containing:
  - **Subscription files** that describe the general subscription settings.
  - **Credentials files (optional)** containing the user name (login) and password for direct connection to the SQL server. These files are encrypted.
  - **Rules/Agreement files (optional),** with the text related to Subscriptions such as descriptions, rules and/or any agreements including the License Agreement, Privacy Agreement, etc.
- **"*Mappings*"** subfolder, with the configuration files that describe mapping between the database tables/columns and the items/fields in Outlook.
- **"*Sync2DB.exe.config*"** file, with the general Sync2DB Synchronizer configuration settings.
- **"License"** subfolder containing the client licenses. These files, with a filename extension ".2dblic", are generated automatically and **must not be changed**.

When the configuration package (".2db" file) is applied, all files are copied into the "*Configuration*" subfolder and all old files are overwritten.

Note: We strongly recommend using the names of the data table columns in the same case as they were used within a table definition (i.e. "CREATE TABLE") command.

3.3 Subscription file

- The subscription file can contain either a detailed definition for one or several subscriptions (this subscription type is called "Local Subscription") or can be very brief and only contain information where to get the detailed subscription definition. This subscription type is called "Remote Subscription". In case of Remote Subscription, the detailed subscription files and mapping files are stored centrally - in the database or in the file system of a server (see "3.10 Remote Subscriptions").
- The Subscription file must have an ".xml" filename extension. Generally, there will be one subscription file for each Subscription but, if necessary, one file may contain several Subscriptions. Use the template subscription files from the *(Administration Tool Application*

folder)\Documentation\Samples\ folder to prepare subscription files. (This is typically *C:\Program Files\4Team Corporation\4Team Sync2DB Administration\Documentation\Samples\*.)

File structure and notes:

- **The order of the XML nodes is important.** Leave them in the same order as they are in the template.
- The subscription file must contain either one node `<localSubscriptions>` or one node <remoteSubscriptions>.
- The file can contain one or several Subscriptions (nodes <subscription>) within node <localSubscriptions>.
- Each subscription can contain one or more Folders (nodes <item>).
- The following settings (nodes) can be present either in the Folder or in the Subscription:
    - <adminConnection>
    - <connection>
    - <authentication>
    - <schedule>
    - <pst>

  If such a node is present in the Folder, i.e. within the node <item>, its value will be used; otherwise, the value of the node from the Subscription, i.e. the parent node <subscription>, will be used.
- <adminConnection> (optional) - License database settings: (The License database is the database where licenses are stored. This database should be selected during Corporate License activation via Sync2DB Activation Wizard). The following options are available: Option 1: Database connection via Web Services.
    - <webServiceAuthentication> (optional) - The value must be the same as specified on IIS settings. The default value when this node is omitted is "*None*".
    - <webService> - the URL address of Web Service. For example:
      <webService>*http://sandbox.sync2db.com/WebServices/FTClientLicenseProviderService.asmx*</webService>
    - <connectionName> (optional) – the name of the connection to the SQL server within the *web.config* file of the Web Service web application. The default value when this node is omitted is "*Default*".

  Option 2: Database connection via direct connect.
    - <connectionString> - Connection string to the SQL server and the License Database (described below).
    - <providerName> - described below.
    - <credentials> - described below.

  Option 3: Use the main database settings (for direct connect within node <connection> only).
    - The same settings specified for the main database will be used if there is no node <adminConnection>. If the license tables are not found, the Subscription will require one of the Professional Licenses (see Option 4a).

  Option 4a: Professional License is required (for direct connect within node <connection>)
    - If there is no node <adminConnection> and the license tables don't exist in the main database, the Subscription requires one of the Professional Licenses. A user must have either a Professional Corporate License or a Professional Private License. The license database is not required in this case.

Option 4b: Professional License is required (for connection via Web Services within node <connection>)

- If there is no node <adminConnection>, the Subscription requires one of the Professional Licenses. A user must have either a Professional Corporate License or a Professional Private License. The license database is not required in this case.

- <connection> - Database connection settings:

  Option 1: Database connection via Web Services.
  - <webServiceAuthentication> (optional) - described above.
  - <webService> - the URL address of Web Service. E.g: <webService>*http://sandbox.sync2db.com/WebServices/FTDbServerSyncProviderService.asmx*</webService>
  - <connectionName> (optional) – the name of the connection to the SQL server within the *web.config* file of the Web Service web application. The default value to be used when the node is omitted is "*Default*".

  Option 2: Database connection via direct connect.
  - <connectionString> - the connection string to the SQL server. For example: <connectionString> - the connection *Server=MY_SQL_SERVER_NAME; Database=MyDatabase; Integrated Security=False;*</connectionString>
    or
    <connectionString>*Server=.; Database=MyDatabase; Integrated Security=True;*</connectionString>.

    Refer to http://connectionstrings.com/ for more detailed information and samples of connection strings for various databases. Pay particular attention to the parameter ***"Integrated Security"***[6]. It defines the type of authentication at the SQL Server level. A "True" value means "**Windows Authentication**" also known as a "Trusted Connection". A "False" value means "**SQL Server Authentication**", in which case the developer must prepare the credentials file using the *Sync2DB Administration Tools* application and specify the credentials filename within the <credentials> node using the filename without the ".data" extension.
  - <providerName> - use "*System.Data.SqlClient*" value for Microsoft® SQL Server, "*MySql.Data.MySqlClient*" value for MySQL® and "*System.Data.OracleClient*" value for Oracle® RDBMS[7]. For example:
    <providerName>*System.Data.SqlClient*</providerName>
  - <credentials> (optional) - the Credential's filename without the "*.data*" extension. This tag is required when the connection string doesn't contain username and password, but they are required (e.g., *SQL Server Authentication* is used for Microsoft® SQL Server, i.e. <connectionString> contains "*Integrated Security=False;*"). The *Sync2DB Administration Tools* application is used to generate the credentials file. The Credentials file must have the filename extension "*.data*" and must be placed within the "*Subscriptions*" subfolder.

- <authentication> - the type of authentication at the Sync2DB level. The valid values are:

[6] For the Microsoft® SQL Server only.
[7] Oracle® RDBMS is not supported in the current version.

  - **None** - No authentication.
  - **Windows** - Windows authentication. A login form will not be displayed if client and server are within one domain. The *spFT_AuthenticateUser* stored procedure must be defined in the database where the data tables are located. The template is included in the SQL script. See "2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers)" for details.
  - **Forms** - Forms authentication. A login form will be displayed prompting the user for user name (login) and password. The *spFT_AuthenticateUser* stored procedure must be defined in the database where the data tables are located. The template is included in the SQL script. See "2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers)" for details.

  The default value when this node is omitted is "*None*".

- <pst> - the name of the PST store. For example, "Archive Folders", "Sync2DB Folders", "Microsoft Dynamics CRM"), or the "**{olDefaultStore}**" keyword for the default PST store. "**{olDefaultStore}**" equates to "Personal Folders" and using this keyword instead of "Personal Folders" is strongly recommended.

Nodes within the <subscription> node:

- <name> - the name of the Subscription. The name must be unique across all Subscriptions in all subscription files. Once set, this name must not be changed.
- <authentication> - described above.
- <enabled> (optional) – indicates whether or not this Subscription is enabled. The value defaults to **"true"** whenever this node is omitted. The value "**false**" indicates that this Subscription is disabled, however the user can enable it at any time via the Synchronizer menus. When a Subscription is disabled:
  - All folders of this Subscription are grayed out in the main Synchronizer's form.
  - Scheduled (automatic) synchronization is not performed for this Subscription.
  - Manual (by user request) synchronization of this Subscription called via the main menu is available to the user.
- <description> - the description of the Subscription. This description will be visible to users in the Sync2DB Synchronizer.
- <order> (optional) - defines the order of Subscriptions in the main Synchronizer form as well as the synchronization order when synchronization of all subscriptions is initiated.
- <schedule> (optional) - defines the schedule of automatic background synchronization for this Subscription. If this node is omitted, only manual (by user request) synchronization will be available. The format is:
  [Days=* | {<week_day>[,...n]};] [Start=*time*;] [End= *time*;] Period=*period*;
  - Values for <week_day> are Sunday, Monday, Tuesday, Wednesday, Thursday, Friday, and Saturday.
  - "Every day" can be specified as "Days=*;" or by simply omitting "Days=;".
  - "*time*" format is "H:mm" in the range "0:00" - "23:59".
  - The default "*Start*" is "0:00"
  - The default "*End*" is "23:59".
  - "*period*" is specified in minutes. This is the only mandatory parameter.

For example:
`<schedule>Days=Monday,Tuesday,Wednesday,Thursday,Friday; Start=8:00; End=18:00; Period=1;</schedule>`
`<schedule>Period=180;</schedule>`
`<schedule>Days=*; End=7:30; Period=5;</schedule>`
`<schedule>Days=Monday,Sunday,Thursday; Start=9:00; End=17:00; Period=15;</schedule>`

- **Rules/Agreement** contains the text for the description, rules, and/or any agreements including the License Agreement, Privacy Agreement, etc that will be shown to the user in the Privacy Alert wizard. This wizard is displayed whenever the Sync2DB configuration package is applied, for example, when the user opens a ".2db" file. The user must accept the Rules/Agreement to apply the configuration package. To use the Rule/Agreement feature, place the text in the `<laText>` node of the subscription file or place the text in a ".rtf" or ".txt" file and specify that filename in the `<laFile>` node. Several Subscriptions can use the same Rules/Agreement file. The Rules/Agreement file must be either a rich text format (RTF) or plain text format (standard ASCII text) file.
    - `<laText>` (optional) – the text of Rules/Agreement. Only plain text (without any formatting) is allowed.
    - `<laFile>` (optional) – the filename of the Rules/Agreement file including the filename extension.

  If both of these nodes are absent or empty, the Privacy Alert wizard will be displayed without the "Rules/Agreement" step.
- `<url>` (optional) – the URL address of the web page that contains additional information for the user about this Subscription. ("More info (Web)" link in the Privacy Alert wizard).

Nodes within the `<item>` node:

- `<name>`[8] - a conditional identification name of the folder. The name must be unique within the current Subscription.
- `<description>` (Not used in the current version) – This node can be used for notes.
- `<direction>` - the synchronization type or direction. The valid values are:
    - **DownloadOnly** - download items from the database to Outlook only
    - **UploadOnly** - upload items from Outlook to the database only
    - **Bidirectional** - two-way data synchronization between Outlook and the database
- `<enabled>` (optional) – indicates whether or not this Folder is enabled. The value defaults to **"true"** whenever this node is omitted. The value "**false**" indicates that this Folder is disabled, however the user can enable it at any time via the Synchronizer menus. When a folder is disabled:
    - This Folder is grayed out in the main Synchronizer's form.
    - Scheduled (automatic) synchronization is not performed for this Folder.
    - Manual (by user request) synchronization of this folder called via the right-click menu in the main form is available to the user.

[8] This node was added in version 1.2.

- <order> (optional) – defines the order of Folder synchronization within a Subscription whenever synchronization is initiated. The order of folders within subscription in the main Synchronizer's form is **always alphabetical** and doesn't depend on this setting.
- <schedule> (optional) – defines the schedule of automatic background synchronization for this Folder (see above for instructions on how to define it).
- <folder> - the full path of the Outlook folder (excluding the root folder) to be synchronized. E.g.:
  <folder>*4Team Sync2DB\Demo - Contacts*</folder>
  or
  <folder>*4Team Sync2DB\Full Outlook Sync\Calendar*</folder>
  Any of these pre-defined constants can also be used:
    - **{olFolderCalendar}** - default Calendar folder (typically "Calendar");
    - **{olFolderContacts}** - default Contacts folder (typically "Contacts");
    - **{olFolderDrafts}** - default Drafts folder (typically "Drafts");
    - **{olFolderInbox}** - default Inbox folder (typically "Inbox");
    - **{olFolderJournal}** - default Journal folder (typically "Journal");
    - **{olFolderNotes}** - default Notes folder (typically "Notes");
    - **{olFolderOutbox}** - default Outbox folder (typically "Outbox");
    - **{olFolderSentMail}** - default Sent Mail folder (typically "Sent Items");
    - **{olFolderTasks}** - default Tasks folder (typically "Tasks");

  It is highly recommended that these pre-defined constants be used for any default folders.
- <folderType> (optional) - the type of folder that is specified in the <folder> node. Although optional, it is highly recommended that this node always be specified. It can only be omitted when the Folder already exists in the user's Outlook. The following values are allowed:
    - **IPF.Appointment** for Calendar folders
    - **IPF.Contact** for Contacts folders
    - **IPF.Note** for Mail and Post folders
    - **IPF.Journal** for Journal folders
    - **IPF.StickyNote** for Notes folders
    - **IPF.Task** for Tasks folders
- <fields> - a comma-separated list of fields that are to be synchronized. For example:
  <fields>Subject,StartDate,DueDate,Status,Body,PercentComplete</fields>
  All fields listed must be mapped in the mapping file (described elsewhere). An "*" can be used if all fields specified in the mapping file are to be synchronized. E.g.:
  <fields>*</fields>
  Note: *Use the names of the* ***Outlook item fields*** *here, not the names of the columns in the data tables!*
- <equivalenceFields> - a comma-separated list of fields used to determine equivalent item-record pairs. During synchronization the Sync2DB Synchronizer compares items in Outlook with records in the database to locate matching pairs. An item and record will be considered identical if the values of all the fields specified in the <equivalenceFields> node match. The developer must specify the <equivalenceFields> node correctly. The rule is to list the fields that uniquely identify each item in Outlook and each record in the data table. Unfortunately, the primary key of the data table cannot be used here, only the fields that exist in Outlook such as Subject, FullName, StartDate, etc. In most cases, the recommended values are:
  For contacts:

<equivalenceFields>FullName,CompanyName,EMail1Address,HomeTelephoneNumber,MobileTelephoneNumber</equivalenceFields>

For tasks:

<equivalenceFields>Subject,StartDate,DueDate</equivalenceFields>

For e-mails:

<equivalenceFields>Subject,SenderEmailAddress,To,CC,BCC,SentOn,ReceivedTime</equivalenceFields>

For notes:

<equivalenceFields>Body</equivalenceFields>

For journal items:

<equivalenceFields>Subject,Type,Start,End</equivalenceFields>

Note:

- *Use the names of the **Outlook item fields** here, not the names of the columns in the data tables!*
- We strongly recommend indexing columns in the database tables that are the members of <equivalenceFields>.

- <table> - the name of the data table. This can be the name of a real table or view (virtual table). **One table name can be used only once**, i.e. the name of the data table must be unique across all subscriptions[9]. To synchronize data in one data table such as *MyContactsTable* with another Folder in one or several Subscriptions, a view for it will need to be created. For example, the SQL command "*CREATE VIEW MyContactsTable_view1 AS SELECT * FROM MyContactsTable*" can be used to create a view whose name can then be specified in the node. One Folder specification will define the node <table>MyContactsTable</table> and the other will define the node <table>MyContactsTable_view1</table>. If more Folders need to be synchronized with *MyContactsTable* table, additional views will need to be created.
- <mapping> - the name of the mapping file without the ".xml" extension. This file should be placed within the "Mappings" folder or any of its subfolders. Several Folders in one or more Subscriptions can reference the same mapping file. See the "3.6 Mapping files." section for more information about mapping.
- <attachment> (optional) - use this node if attachments to an item should be synchronized.
  - <direction> - use the same value that was specified in the <direction> node of the parent (Folder) node.
  - <table> - the name of the database table or view where attachments are stored. (Attachments should be stored in a separate database table).
  - <mapping> - the name of the attachment's mapping file without the ".xml" extension. This file should be placed within the "Mappings" folder or any of its subfolders. The same mapping file can be specified for the attachments of several Folders defined in one or more Subscriptions. Use the file *(Administration Tool Application folder)\Documentation\Samples\MSSharePoint\Mappings\SP_Attachments.xml* as a template. See the "3.6 Mapping files." section for more information about mapping.
- <trackingFields> - a comma-separated list of the tracking columns as described in the "2.3 Metadata " section. These seven (7) items should be listed in the strict order as shown here:

<trackingFields>tp_Author,tp_Created,tp_Editor,tp_Modified,ft_Editor,ft_Deleted,FT_ClientID</trackingFields>

[9] Restriction in the current version.

If deleting operations are not allowed, the 5th and 6th items must be present but empty as:
<trackingFields>ft_InsertId,ft_InsertAnchor,ft_UpdateId,ft_UpdateAnchor,,,ft_ClientID</trackingFields>
If the SQL scripts recommended in the "2.4 Adding metadata objects (change tracking columns, tombstone table and tracking triggers)" section were used, the last line of the execution report from the *spFT_AddMetadataColumns* stored procedure will contain a ready-to-use value for the<trackingFields> node. Simply copy it from there. ***Note:*** *Use the names of the* ***columns in the data tables*** *here, not the names of the Outlook item fields!*

- <getPrimaryKey> (optional) – the name and parameters of the stored procedure used for primary key generation. E.g.:
  <getPrimaryKey>spFT_GetNextId @paramListID='8000000D-BA76-4459-9223-D48E11AA8AD5'</getPrimaryKey>
  Omit this node when the primary key column is automatically numbered.
  Explanation: If the direction of folder synchronization is "Upload only" or "Bidirectional", new items in the Outlook folder should be inserted as new records in the data table. Therefore, it is important for the developer to know how the new values of the primary keys are generated. The current version supports the following methods of primary key generation:
  - **Automatic** when the primary key column is described as IDENTITY (AUTO_INCREMENT) in the table definition.
  - **Custom, using Stored Procedure** must be used in all other cases.

  The name and parameter values, if required, must be specified in the <getPrimaryKey> node of the Subscription configuration file for each respective table. This stored procedure must return a one-row record set with columns that are members of the defined primary key. Values of the resulting record set will be used by the Synchronizer during the "insert" command for columns that are members of the primary key. For an example refer to the SQL script file *SharePoint_CreateSync2DBObjects.sql* in the *(Administration Tool Application folder)\Documentation\Samples\MSSharePoint\MSSQL Scripts* folder.
- <deleteCommand> (optional) – the name of the stored procedure used for record deletion in the database. Omit this node when record deletion is performed by the simple SQL "DELETE" command. See sample SQL scripts at "Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

- Microsoft® SharePoint"
- <clientFilter> (optional) - allows for the filtering of items in the Outlook folder before synchronization. This node has the same definition as the <serverFilter> node defined below.
  Note: *Use the name of the* ***Outlook item field*** *here, not the name of the column in the data table!*
- <serverFilter> (optional) - allows records to be filtered in the data table before synchronization. See sample SQL scripts at "Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

- Microsoft® SharePoint"

- `<column>` - column name.
  Note:
  *Use the name of the **column in the data table** here, not the name of the Outlook item field!*
  All columns listed must be mapped in the mapping file!
- `<type> (optional)` - column type. For more information about types, see the "3.6 Mapping files." section. The default value when the node is omitted is **String**.
- `<operation> (optional)` - the allowed values are:
  - Equals
  - NotEquals
  - Greater
  - GreaterOrEquals
  - Less
  - LessOrEquals
  - Like
  - NotLike
  - IsNull
  - IsNotNull

  The default value when the node is omitted is "Equals". See sample SQL scripts at "Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

Microsoft® SharePoint"

- `<value>` - filter value (any text). This can also be one of the following keywords:
  - **ClientID** - the unique id of the Sync2DB client (i.e. combination of a Windows user and a computer) generated automatically when the first time the user starts the Sync2DB Synchronizer. This value will never change, once set.
  - **UserID** – the unique id of the Sync2DB user in the database returned by the **spFT_AuthenticateUser** stored procedure.

  ***Note:** Sync2DB user and Sync2DB client are not the same!* See sample SQL scripts at "Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

Microsoft® SharePoint"

3.4 Credentials file

The credentials file is an encrypted file that contain the user name (login) and password needed for a direct connection to the SQL server (`<connectionString>` node contains "*Integrated Security=False;*"[10]). In this case, the developer must prepare a credentials file. The credentials file is not needed in the following cases:

[10] For Microsoft® SQL Server only.

1. When "Windows Authentication", also known as a "Trusted Connection", is used for a direct connection to the SQL server (<connectionString> node contains "*Integrated Security=True;*").
2. When Web Services (not a direct connect) are used for database access (the set of settings "<webService>, <connectionName>" is used but not the set "<connectionString>, <providerName>, <credentials>").

The Sync2DB Administration Tools application should be used to generate a credentials file. This application generates the file "*Credentials.data*" and places it into the Sync2DB Administration Tools User Application Data folder. Typically this is *C:\Documents and Settings\(WINDOWS_USER_NAME)\Application Data\4Team\ForTeam.Client.Administration\Data\* for a Windows XP system and *C:\Users\(WINDOWS_USER_NAME)\AppData\Roaming\4Team\ForTeam.Client.Administration\Data\* for a Windows Vista system.

Copy this file into the "Subscriptions" folder and rename it so that it reflects the Subscription it is related to. E.g., the target name can be "*Credentials_SharePoint.data*". Do not change the filename extension since the credentials file must have the ".data" filename extension.

3.5 Rules/Agreement file

The Rules/Agreement file contains text related to the Subscription. It may contain descriptions, rules and/or any agreements including a License Agreement, a Privacy Agreement, etc. The agreement text will be shown to the user in the Privacy Alert wizard displayed when the Sync2DB configuration package is deployed (whenever the user opens a ".2db" file). The user must accept the Rules/Agreement in order to deploy the configuration package. The name of the Rules/Agreement file is specified in the subscription file. Several Subscriptions can use the same Rules/Agreement file.

The Rules/Agreement file must be a rich text format (RTF) or plain text format (standard ASCII text) file. Prepare the ".rtf" or ".txt" file with the Rules/Agreement, place it in the "Subscriptions" folder, and specify the name of this file (with the filename extension) in the <laFile> node of the subscription file. For example:

```
<laFile>Sync2DBDemo1SandboxPrivacyPolicy.rtf </laFile>
```

An alternative to using the Rules/Agreement file is to place the Rules/Agreement text directly in the subscription file using the <laText> node. Only plain text without any formatting is allowed here. If both the <laFile> and <laText> nodes are absent or empty, the Privacy Alert wizard will be displayed without the "Rules/Agreement" step.

3.6 Mapping files.

The mapping files are configuration files that describe mapping between tables/columns in the database and items/fields in Outlook. Each mapping file describes mapping of one Outlook item for one or several Subscriptions. The name of the file is specified in the subscription file and must have an ".xml" filename extension. Several Folders in one or more Subscriptions can reference the same mapping file.

The Sync2DB Synchronizer searches for the specified file in all subfolders under the "Mappings" folder. Therefore it is recommended that a separate subfolder be created for each Subscription and that all related mapping files be placed there. For example, all mapping files related to the predefined demo "4Team Demo Sandbox" Subscription are placed in the "*\Configuration\Mappings\Demo1Sandbox\*" folder.
Please note that the name of the mapping file must be unique across all subfolders of the "Mappings" folder (i.e. "*TaskItem.xml"* cannot exist under both "*\Mappings\Subscription1\*" and "*\Mappings\Subscription2\*").

To prepare a mapping file, use one of the sample mapping files from the *(Administration Tool Application folder)\Documentation\Samples* folder as a template. Typically this is *C:\Program Files\4Team Corporation\4Team Sync2DB Administration\Documentation\Samples\*.

Mapping templates with all of the Outlook properties are placed in the *(Administration Tool Application folder)\Documentation\Samples\Templates\* folder.

File structure and notes:

- **The order of various XML nodes is important**. Please make sure they are ordered in the same sequence as defined here.
- The mapping file contains definitions for pairs of "database column to Outlook item field". (<mapping> nodes).

Nodes within the <mapping> node:

- <Active> (optional) – the default value when this node is omitted is **true**. The Sync2DB Synchronizer will ignore this <mapping> node if the value is **false**.
- <PropertyName> - the name of the Outlook item field (also known as the Outlook item ***property***). For clarity on the meaning of **property**, refer to the lists of Outlook Fields and Equivalent Properties:
    - Outlook 2003: http://msdn.microsoft.com/en-us/library/aa204691(office.11).aspx
    - Outlook 2007: http://msdn.microsoft.com/en-us/library/bb176447.aspx

  Note:
    - **Property names must be unique across the current mapping file**. Each Outlook property in a given Outlook folder can only be mapped to one database column. But one database column can be mapped to several Outlook properties.
    - **If the specified property cannot be found** in Outlook during synchronization, a new Folder User Property will be created with this name. (This new property is also known as a "Custom field" or "User-defined field in Folder").
    - **Property names can be fictitious**. This is the case when both the <CanRead> and <CanWrite> nodes (see below) are specified as **false**. In this case, the Sync2DB Synchronizer will not attempt to read or write this property in Outlook. As an example, **primary keys** are typically defined as fictitious properties since **they must be defined in the mapping** for proper synchronization but do not actually map to a real Outlook property. These are mapped as: <CanRead>false</CanRead> and <CanWrite>false</CanWrite>.
- <CanRead> (optional) - the default value when this node is omitted is **true**. The Sync2DB Synchronizer will not read this property from Outlook when the value is **false**.

- <CanWrite> (optional) - the default value when this node is omitted is **true**. The Sync2DB Synchronizer will not write this property to Outlook when the value is **false**.
- <ColumnName> (optional) - the name of the column in the database data table. Omit this node when the default Outlook property value (or "client constant") is used instead of specifying the Outlook property<=>database column mapping. See the bottom of this topic for more info.
- <PropertyType> - the data type of the Outlook item property. **Leave the same values that were in the template.** If the used template doesn't contain this property, refer to templates with all the Outlook properties located in the *(Administration Tool Application folder)\Documentation\Samples\Templates\* folder.
- <ColumnType> (optional) - the only case when this node is to be used is when the datetime field is processed as date (without time). The value should be "DATE" in this case (<ColumnType>DATE<ColumnType>).
- <Length> (optional) - the maximum length of the data table column with a string data type. Omit this node for all other data types. For example: a data table column defined as `VARCHAR(255)` would be defined as <Length>255</Length>.
- <AllowNull> (Not used in the current version) – this node can be omitted.
- <Value> (optional) – the default value for the "client constants" and "server constants" (any text). Omit this node if the standard Outlook property<=>database column mapping is used. This can also be one of the following keywords:
  - **ClientID** – indicating the unique id of the client (i.e. combination of a Windows User and a computer name) that is generated automatically upon Synchronizer installation and remains constant thereafter.
  - **UserID** – indicating the unique id of a user in the database (returned by the **spFT_AuthenticateUser** stored procedure).
  - **GETDATE()** – indicating the current date and time (for "server constants" only).
  - **GETUTCDATE()** – indicating the current date and time in UTC (for "server constants" only).

  See sample SQL scripts at "Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

Microsoft® SharePoint"

- <PrimaryKey> (optional) – the default value when this node is omitted is **false**. Specify **true** for properties that are mapped to primary key columns in cases where problems with the retrieval of data scheme occur. E.g., if the Sync2DB Synchronizer cannot synchronize a particular complex view, set the node to **true** for properties that are mapped to the unique columns of the view.
- <MapTo> (optional) - allows enumerated fields to be mapped correctly in cases where Outlook enumerators are different from the database values.

  Example 1: To map the Outlook *Importance* property to the string column *UserPriority* in the data table (which can take on the these values: "LOW", "NORMAL", "HIGH", "VIP" and NULL) specify:

```
<mapping>
    <PropertyName>Importance</PropertyName>
    <ColumnName>UserPriority</ColumnName>
    <PropertyType>OlImportance</PropertyType>
```

```
        <ColumnType>NVARCHAR</ColumnType>
        <Length>20</Length>
        <MapTo>olImportanceNormal : NORMAL, olImportanceHigh : HIGH,
      olImportanceHigh : VIP, olImportanceLow : LOW</MapTo>
      </mapping>
```

Note: The first value in the list is taken as the default. The default will be used if the value of the *UserPriority* column in the data table is not found in the list during the download to Outlook. In this example, *olImportanceNormal* will be used for records where *UserPriority* is set to NULL or "NOT_DEFINED".

Example 2: To map the Outlook Task *Status* property to the BIT column *TaskDone* in the data table (which can be 0 or 1) specify:

```
    <mapping>
        <PropertyName>Status</PropertyName>
        <ColumnName>TaskDone</ColumnName>
        <PropertyType>OlTaskStatus</PropertyType>
        <ColumnType>BIT</ColumnType>
        <MapTo>olTaskNotStarted : False, olTaskComplete : True</MapTo>
      </mapping>
```

After downloading to Outlook, records with *TaskDone*=0 will appear in Outlook as tasks with the Status of "Not Started" and records with *TaskDone*=1 will appear in Outlook as tasks with the Status of "Completed". After uploading from Outlook, tasks with the Status "Completed" will appear in the database with *TaskDone*=1 and tasks with all other types of Status ("Not Started", "In Progress", "Deferred", "Waiting on someone else") will appear in the database with *TaskDone*=0.

To find the correct property types and values of Outlook enumerators, please refer to the List of Outlook object model constants at http://support.microsoft.com/kb/285202.

- <DateTimeMode> (optional) - defines the time mode (**"Local"** or **"Utc"**) for the data table columns with a *datetime* data type. Set the node to **"Utc"** if the *datetime* values within this database column are stored in UTC form and set the node to **"Local"** if the values are stored in local time. The default value when the node is omitted is **"Utc"**. Omit this node for all other data types.
- <Indexed> (Not used in the current version) - This node can be omitted.
- <Description> (Not used in the current version) – This node can be used for your own notes.

Note:

-A mapping must be specified for each primary key column. If one of these columns cannot be mapped to a real Outlook property, which is quite typical, specify this property as fictitious using <CanRead>false</CanRead> and <CanWrite>false</CanWrite>.

-In addition to using a mapping to map an Outlook property to a database column, a mapping can be used to define the default value (or "client constant") of the Outlook property during download or to define the default value of the database column (or "server constant") during upload. To make a "client constant" specify the <Value> node but omit the <ColumnName> node.

Example 1: The following example will assign a Category to Outlook items during download:

```
<mapping>
      <PropertyName>Categories</PropertyName>
```

```
        <Value>Sync2DB SharePoint</Value>
</mapping>
```

Example 2: Often it is convenient to mark all downloaded items as Un-Read. Use the following “client constant” for this purpose:

```
<mapping>
        <PropertyName>UnRead</PropertyName>
        <Value>true</Value>
</mapping>
```

-If the <clientFilter> node in a Bidirectional subscription is used, don’t forget to specify the “client constant” with the same value, otherwise downloaded items will not upload afterwards due to filter restrictions.

-To make a “server constant”, you should specify the <Value> and <ColumnName> nodes but set both <CanRead> and <CanWrite> values to **false**. The value of the <PropertyName> node can be anything because it will be a fictitious property.

Example:

```
<mapping>
        <PropertyName>Dummy1</PropertyName>
        <CanRead>false</CanRead>
        <CanWrite>false</CanWrite>
        <ColumnName>RecordModified</ColumnName>
        <Value>GETUTCDATE()</Value>
</mapping>
```

-In the current version the following restrictions exist:

- Properties in the mapping file must be placed in the correct order. E.g., if one database column is mapped to several Outlook fields (one time as read-write and the other as write-only), it is mandatory to place the property section with the read-write mapping first.
- Property sections for different Outlook fields should also be in the correct order.
- For "DownloadOnly" and "Bidirectional" synchronization with MySQL® database, the *ft_Reserved* column must be specified using the following “server constant”:

```
<mapping>
        <PropertyName>ft_Reserved</PropertyName>
        <CanRead>false</CanRead>
        <CanWrite>false</CanWrite>
        <ColumnName>ft_Reserved</ColumnName>
        <ColumnType>INT</ColumnType>
        <Value>1</Value>
</mapping>
```

- For "DownloadOnly" and "Bidirectional" synchronization, the *MessageClass* property must be specified using the following “client constant”:

```
<mapping>
        <PropertyName>MessageClass</PropertyName>
        <Value>IPM.Task</Value>
</mapping>
```

  where valid values are:
  - **IPM.Appointment** for Calendar items
  - **IPM.Contact** for Contacts
  - **IPM.DistList** for Distribution Lists

- **IPM.Document** for Documents
- **IPM.Note** for Mails
- **IPM.Post** for Post items
- **IPM.Activity** for Journal entries
- **IPM.StickyNote** for Notes
- **IPM.Task** for Tasks

3.7 Sync2DB.exe.config file

The target links of the “Corporate Site” and "Corporate Help" menu items in the “Help” menu of the Synch2DB Synchronizer can be customized. To do this, find the following lines in the Sync2DB.exe.config file and change the text contained in the “value” section appropriately:

```
<add key="CorporateSite" value="http://www.sync2db.com/"/>
<add key="CorporateHelp" value="http://www.sync2db.com/Help_Center.asp"/>
```

Any valid URL address or filename (with path) can be specified. E.g., a file with a “.doc”, “.rtf”, “.txt”, “.pdf”, or “.htm” file extension. When the user clicks on that menu item, the specified URL or file will open via the associated registered program (i.e. Internet Browser, Microsoft® Word, etc). When any of these options is omitted, the corresponding menu item in the “Help” menu will be absent.

The default e-mail address that is used by the “Send Error Log” function (“Help” menu of the Synch2DB Synchronizer), can be customized. To do this, insert the following line in the Sync2DB.exe.config file (section `<appSettings>`):

```
<add key="BugReportEmail" value="your@email.com"/>
```

The default value when this option is omitted is support@sync2db.com.

3.8 Making the Configuration package (“.2db” file)

To prepare the configuration package into a “.2db” file, compress the prepared subfolders of the “*Configuration*” folder (“*Subscriptions*” and “*Mappings*”) and the “*Sync2DB.exe.config*” file into a ZIP archive and rename the resulting “.zip” file to have a “.2db” extension. The configuration package is ready. **Don’t forget to test the prepared configuration package before sending it to other users.**

3.9 Distributing the configuration package (“.2db” file)

The Configuration Package can be distributed in several ways:

- As an attachment (“.2db” file) in an e-mail.
- As a link to the “.2db” file on a web page or in an e-mail.
- Using a remote (or administrative) installation. Administrators can copy the configuration files (not zipped to a “.2db” file) directly to the *4Team\Sync2DB\Configuration\* folder in the user’s Application Data folder on the user’s computer. Typically this is *C:\Documents and Settings\(WINDOWS_USER_NAME)\Application Data\4Team\Sync2DB\Configuration\* for Windows XP systems and *C:\Users\(WINDOWS_USER_NAME)\AppData\Roaming\4Team\Sync2DB\Configuration\* for Windows Vista systems.

When the user opens the ".2db" file, the Privacy Alert wizard regarding the included Subscriptions and consisting of 2 steps is shown. The first step contains the rules and license agreement and the second shows the list of Outlook folders that will be synchronized. The user must accept both of these in order to synchronize the folders included in the Subscriptions. If the user doesn't accept both steps of the Privacy Alert wizard, the configurations in the currently opened ".2db" file **will not be deployed**.

The Privacy Alert wizard is also shown:

- When the first time the user starts the Sync2DB Synchronizer after an Administrator copies the configuration files directly in his Application Data folder.
- When using the Sync2DB Synchronizer installation package with the "demo" option which includes the free predefined demo Subscription- *the 4Team Demo Sandbox*.

In all cases, if the user doesn't accept both steps of the Privacy Alert wizard, all of the unaccepted subscription files **will not be loaded and used**. The Privacy Alert wizard **will appear again during the next start** of the Sync2DB Synchronizer.

3.10 Remote Subscriptions

The Subscription files and Mapping files can be stored centrally - in the database as well as on the file system of a server. This is called "Remote Subscriptions". See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\Subscriptions\MySQLAndRemote* folder for a sample. The file *DS_SubscriptionsRemote.xml* is a sample of the local (which should be placed on the user's computer) subscription file. It can be the only configuration file on the user's computer. All other configuration files, including subscription and mapping files, are stored on the server. The structure of these files is the same as for locally stored subscription and mapping files.

There are 2 locations where to store configuration files on the server:

- The file system.
  In this case the Sync2DB Web Services must be present and configured on the server. The key "SubscriptionLocation" within the *web.config* file must have the "FS" value (see the file *(Administration Tool Application folder)\ Web Services\web.config*). The key "AppConfigurationFolder" defines the root folder for configuration files. This folder should contain 2 subfolders - "Mappings" and "Subscriptions".
- The database.
  In this case the Sync2DB Web Services can be used. The key "SubscriptionLocation" within the *web.config* file must have the "DB" value.
  Please note, that the current version of Sync2DB uses the "Default" connection string only (within the *web.config* file) to retrieve configuration files from a database.
  You can do without Web Services - specify the set of nodes `<connectionString>,<providerName>,<credentials>` within a local subscription file instead of node `<webService>` to use the direct connection to the database.
  Please note, that to use this option, you should create the table ftSync2DBSubscriptions (see script file *Custom_CreateObjects.sql*)

4. Included samples

Demo sandbox

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\* folder.

Demo sandbox (MySQL®)

See files in the *(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\MySQL Scripts* folder.

Microsoft® SharePoint

See files in the *(Administration Tool Application folder)\Documentation\Samples\MSSharePoint\* folder.

Online Store

See files in the *(Administration Tool Application folder)\Documentation\Samples\OnlineStore\* folder.

Remote Subscription

See files in the
(Administration Tool Application folder)\Documentation\Samples\Demo1Sandbox\Subscriptions\MySQLAndRemote
folder.

Mapping Templates

Mapping templates (mapping files where all Outlook properties are present) are located in the *(Administration Tool Application folder)\Documentation\Samples\Templates\* folder.

More samples

Also, see the *Administration Tool Application folder)\Documentation\Sync2DBCases.doc* file for more samples.

5. Installation recommendations

5.1 Sync2DB Synchronizer installation

Remote (or Administrative) installation is recommended for the Sync2DB Synchronizer. See http://msdn.microsoft.com/en-us/library/aa367541.aspx for details.

5.2 Web Services installation and setup

Steps to perform:

1. Download the Sync2DB Web Services installation package (http://www.sync2db.com/download/Sync2DBWeb_Setup.msi) and install it. During installation a new Web Site or new Virtual Directory will be created.
2. Configure the just created Web Site (Virtual Directory)
3. If the value "Windows" is used in the `<authentication>` node of the Subscription, configure the Web Site (Virtual Directory) to use "Integrated Windows authentication".
4. Modify the connection string properly. (`<add name="Default"...` node within the `<connectionStrings>` node of the web.config file).

5. If connections are to be made to multiple databases, add additional connections with different names. Specify the name within the `<connectionName>` node(s) of the Subscription.
6. Specify the full URL of the Web Services for synchronization within the `<webService>` node of the Subscription. (*FTDbServerSyncProviderService.asmx* is responsible for synchronization). For example:
<webService>*http://sandbox.sync2db.com/WebServices/FTDbServerSyncProviderService.asmx*</webService>
7. If access to the License Database via the Web Services is needed, specify the full URL of the Web Services for licensing within the `<adminConnection> => <webService>` node of the Subscription. (*FTClientLicenseProviderService.asmx* is responsible for licensing). For example:
<webService>*http://sandbox.sync2db.com/WebServices/ FTClientLicenseProviderService.asmx*</ webService>
8. In case when a MySQL® database is used, the MySQL Connector/Net should be installed on the server. Download it from http://dev.mysql.com/downloads/connector/net/5.2.html.

6. System requirements

6.1 Client Environment (for Sync2DB Synchronizer)

Operating System	Microsoft Windows XP Service Pack 2, Microsoft Windows Vista, Microsoft Windows Server 2003, Microsoft Windows Server 2008
Microsoft Outlook	Microsoft Outlook 2002 (Outlook XP) Microsoft Office Outlook 2003 Microsoft Office Outlook 2007
Environment	Microsoft .NET Framework 2.0 (automatically downloaded and installed by the install package)
Hardware	512 MB RAM minimum (recommended RAM is 1 GB or more), 1 GB hard disk space

6.2 Databases (RDBMS)

Microsoft SQL Server	Version 2000 or later, including all editions as well as MSDE
MySQL	Version 5.1 or later

6.3 Server Environment (in case when Sync2DB Web Services are used)

Operating System	Microsoft Windows Server 2003, Microsoft Windows Server 2008
Environment	Microsoft Internet Information Services (IIS) version 6.0 or later,

	Microsoft .NET Framework 2.0
Additional environment (in case when a MySQL database is used)	MySQL Connector/Net version 5.2

7. Additional Resources

4Team Corporation Resources:

- Sync2DB Help Center (http://www.sync2db.com/Help_Center.asp)
- Sync2DB Frequently Asked Questions (http://www.sync2db.com/About.asp#faq)
- Sync2DB Forum (http://forum.sync2db.com/)
- Contact Us (http://www.sync2db.com/Contact.asp)

Other Resources:

- Connection strings for various databases with lots of examples: http://connectionstrings.com/
- Administrative (or remote) installation: http://msdn.microsoft.com/en-us/library/aa367541.aspx
- Outlook Fields and Equivalent Properties (How do I know the Outlook property name?):
    - For Outlook 2003: http://msdn.microsoft.com/en-us/library/aa204691(office.11).aspx
    - For Outlook 2007: http://msdn.microsoft.com/en-us/library/bb176447.aspx
- List of Outlook object model constants: http://support.microsoft.com/kb/285202