AGENDA

GROUP CALENDARING MADE EASY

# AgendaX Database Documentation V6.0


Release 6.0, Revision 2

DROLLINGER TECHNOLOGIES LLC
Bibersteinerstrasse 80
5022 Rombach
Switzerland

Tel: +41 32 512 3103
E-mail: support@agendax.net

Supplied and Supported in the UK by:

essential
Enterprise Messaging Essentials
Channel Court
Hill Road
Clevedon, BS21 7NB

Tel: +44 01275 343199
Fax: +44 01275 340974
E-mail: info@essential.co.uk

# Chapter 1: Tables

AgendaX uses mainly 3 tables for storing meeting data, and a table for data interchange between the AgendaX Web application and the AgendaX Update Service for creating / deleting / modifying meetings. There are other tables, but they are used to store profiles, personal group definitions, and other user data.

The 3 main tables are: AgendaXConfig, AgendaXTable1 and AgendaXTable2.

## 1.1 AgendaXConfig table

AgendaXConfig stores the currently active table (AgendaXTable1 or AgendaXTable2). The Service changes this value at the end of a scan cycle to the table it used to store the newly fetched data from Exchange. At the beginning of a scan cycle, the currently not active table is cleared and during the scan cycle filled with data as it is being read from Exchange.

```
CREATE TABLE [dbo].[AgendaXConfig](
      [ActiveTable] [varchar](50) NULL
) ON [PRIMARY]
```

## 1.2 AgendaXTable1 / AgendaXTable2 tables

The AgendaXTable1 and 2 tables share the same structure. One of the 2 tables may not contain all of the data, as it may be currently used by the AgendaX Update Service to write data that is read from Exchange Server.

An application should therefore always first read AgendaXConfig to know which table can be safely read. The table stored in AgendaXConfig contains the data that was read during the last AgendaX Update Service cycle and contains a complete set of data.

```
CREATE TABLE [dbo].[AgendaXTable1](
      [Name] [varchar](50) NOT NULL,
      [Start] [datetime] NOT NULL,
      [EndTime] [datetime] NOT NULL,
      [Title] [varchar](250) NULL,
      [Status] [smallint] NOT NULL,
      [Location] [varchar](50) NULL,
      [Phone] [varchar](50) NULL,
      [Office] [varchar](50) NULL,
      [Initials] [varchar](50) NULL,
      [ACL] [text] NULL,
      [Category] [varchar](50) NULL,
      [DL] [varchar](250) NULL,
      [Organizer] [varchar](50) NULL,
      [Attendees] [text] NULL,
      [Datasource] [varchar](50) NULL,
      [TimeDifference] [numeric](2, 1) NULL,
      [Alias] [varchar](50) NULL,
      [Department] [varchar](50) NULL,
      [Reserved3] [varchar](50) NULL,
      [AGXBookingNumber] [varchar](250) NULL,
      [ProfTitle] [varchar](50) NULL,
      [OffFax] [varchar](50) NULL,
      [HomePhone] [varchar](50) NULL,
      [MobilePhone] [varchar](50) NULL,
      [AdrComment] [text] NULL,
      [ACLw] [text] NULL,
      [AssocNTAcc] [varchar](50) NULL,
      [OrigAGXBookingNumber] [varchar](50) NULL
      … (custom fields)
) ON [PRIMARY] TEXTIMAGE_ON [PRIMARY]
```

The tables may contain additional columns at the end that are used for custom Outlook fields that can be used in Outlook forms, or, for example the BodyText of Appointments:

```
[ApptText] [varchar](250) NULL
```

This depends on the definition in the configuration panel AgendaXCfg.exe (Add'l fields).

| | |
|---|---|
| Name | Mailbox display name |
| Start | Start time of a meeting |
| EndTime | End time of a meeting |
| Title | Title (Subject) of a meeting. For (Status=9) records, contains the SMTP address of the mailbox. |
| Status | Meeting status (0=free;1=tentative;2=busy;3=out of office) |
| Location | Location of a meeting |
| Phone | Phone number |
| Office | Office location |
| Initials | Initials |
| ACL | Semicolon-delimited list of users (DOMAIN\username) that have read access to this users calendar folder |
| Category | Double forward slash delimited list of Categories that are set on the meeting (e.g. //Favorites//Business//) |
| DL | Double forward slash delimited list of global AgendaX groups the user is a member of (e.g. //Sales//Marketing//) |
| Organizer | Mailbox display name of the Organizer of the meeting |
| Attendees | Semicolon delimited list of meeting attendees mailbox display names (e.g. Adrian Drollinger; John Doe) |
| Datasource | Only used in multi-site setups, specifies the datasource name of the database where the data for the user is stored ($local if data is stored in the current (local) database) |
| TimeDifference | hours of time difference between user and AgendaX server |
| Alias | Mailbox Alias name |
| Department | Department |
| Reserved3 | Reserved for future use |
| AGXBookingNumber | internal AgendaX Booking number for meetings booked in the AgendaX web interface |
| ProfTitle | Job Title |
| OffFax | Office Fax number |
| HomePhone | Home phone number |
| MobilePhone | Mobile phone number |
| AdrComment | Address comment field (for Status=9 records), otherwise the start and end display time zones of the meeting, separated by //, if the multiple time zones feature is enabled, e.g. //W. Europe Standard Time//W. Europe Standard Time// |
| ACLw | Contains the GUID of the meeting in Exchange Server, or, for (Status=9 records), a semicolon-delimited list of mailboxes that have write access to the mailbox calendar (e.g. Adrian Drollinger; John Doe) |
| AssocNTAcc | Windows account associated with this mailbox (DOMAIN\username) |
| OrigAGXBookingNumber | Contains the original AgendaX booking number of the meeting when it was first booked |

There is a record with (Name='$maintenance' and Status=8) that contains in the Start and EndTime values the date and time the service completed the last scan cycle.

Each scanned mailbox has a record with (Status=9) that contains the full mailbox info (Phone numbers, Department, ACL lists, etc.)

All other records are meeting records, where each record contains a meeting. Meetings that span multiple days are split into separate day records. E.g. a meeting from 10/10/2012 15:00 – 12/10/2012 16:00 is split up into the following records:

10/10/2012 15:00:00 – 10/10/2012 23:59:00
11/10/2012 00:00:00 – 11/10/2012 23:59:00
12/10/2012 00:00:00 – 12/10/2012 16:00:00

Recurring meetings are split into each occurrence, each occurrence making up a record in the table.

## 1.3 Meetings table

The Meetings table serves as an interface for booking / modifying / deleting meetings in Exchange Server (Outlook).

| | |
|---|---|
| Name | Windows- Name of the user that books the meeting in the form DOMAIN\Username |
| Start | start time of the meeting |
| EndTime | end time of the meeting |
| Title | subject of the meeting |
| Status | 0=free, 1=tentative, 2=busy, 3=out of office |
| Location | location of the meeting |
| Phone | Reminder- time in minutes |
| Office | unique number, must be different for each meeting booked |
| Initials | 'invitation' or 'no invitation': determines if an Outlook meeting request is sent to the user or if the meeting is directly booked into his calendar |
| ACL | bodytext of the meeting (notes) |
| Category | category of the meeting |
| DL | empty |
| Organizer | organizer of the meeting (only for meetings, for which an Outlook meeting request is sent) (Exchange Display Name) |
| Attendees | meeting attendees (Exchange Display Names, separated by semicolon) |
| TimeBooked | leave this field empty, it will be filled by the Agenda/X Update Service |
| BookingStatus | 1 (=to be booked) 2 (=booked) 3 (=to be deleted) 4(modified, old record) 5 (modified, new record) |
| Reserved1-3 | leave empty |
| ACLw | leave empty |

To add a meeting, just create a record with the above fields filled in as explained and set its BookingStatus to 1 (to be booked).

To delete a meeting, simply set the record's BookingStatus field to 3 (to be deleted).

If a meeting is modified, its BookingStatus value is changed from 2 (booked) to 4 (modified). The rest of the record is kept the same. A new record with the new data of the meeting and with BookingStatus = 5 is then added. It must have the same unique number like the old record (Office field).

## 1.4 Stored procedures

Stored procedures can be executed both at the beginning and the end of a scan cycle. They can be specified in the AgendaX Configuration file AgentX.ini as follows:

```
[Config]
StoredProcStart=t_sql_statement
StoredProcEnd=t_sql_statement
```

T_sql_statement can be any valid T-SQL statement, e.g. exec stored_proc_name

## 1.5 AGXLinkedUsers Table to match remote and local usernames

*If your Exchange Server is hosted in an **untrusted domain**, by an Exchange Hosting Provider, or you use **Office 365 (Exchange Online)**, AgendaX cannot map mailboxes on the foreign domain to local domain accounts. Since AgendaX needs this mapping with the **Full Security** option to determine which user has access to which mailboxes, you need to create associations between mailboxes and local user accounts. These associations can be made in the **AGXLinkedUsers** database table. This table consists of 2 columns, **AGXMailboxName** and **AGXLocalAccount**. Add a row for each user and specify its local user account as follows:*

| | |
|---|---|
| **AGXMailboxName** | Primary SMTP- Address of the mailbox |
| **AGXLocalAccount** | DOMAIN\username of the user in the local domain |

This task can be automated by using a StoredProcEnd- Definition that is executed at the end of each scan cycle like described in the paragraph above.
An example of such a definition, where SMTP- Addresses are matched to MYDOMAIN\Alias in the AGXLinkedUsers Table can be found below:

```
[Config]
StoredProcEnd=TRUNCATE TABLE AGXLinkedUsers;DECLARE @WhichTableIsActive varchar(50) SELECT @WhichTableIsActive = ActiveTable FROM
AgendaXConfig IF @WhichTableIsActive = 'AgendaXTable1' INSERT INTO AGXLinkedUsers(AGXMailboxName, AGXLocalAccount) SELECT Title, 'MYDOMAIN\'
+ Alias FROM AgendaXTable1 WHERE (Status = 9) ELSE INSERT INTO AGXLinkedUsers(AGXMailboxName, AGXLocalAccount) SELECT Title, 'MYDOMAIN\' +
Alias FROM AgendaXTable2 WHERE (Status = 9)
```

Please note that you will always need **two** scan cycles for the table to be completely filled with all users.

## 1.6 Appending tables

Tables can be appended at the end of a scan cycle by defining:

```
[Config]
AppendTable=name_of_table_to_append
```

The appended table has to have the same structure like the AgendaXTable1 and 2 tables.

## 1.7 Backing up the database

Since AgendaX uses a SQL- Database with the 'Simple'- Recovery Model, transaction logs are automatically truncated by SQL in regular intervals, without the need to backup and truncate tables regularly.

If you wish, you may want to back up the following tables:

| | |
|---|---|
| AGXActions, AGXRules, AGXConditions, AGXMasterCategoryList | If Rules are defined and activated (see Chapter 7 in the AgendaX Installation Guide) |
| AGXBookingLog | If the booking log feature is enabled (AgendaXCfg.exe / Log options) |
| AGXFavorites | If the Search Favorites feature is enabled (Administration page / Feature settings) |
| AGXUsers | If you use AgendaX with its own user management system (Chapter 8 in the AgendaX Installation Guide) |
| **Groups** | **If your users save Personal groups** |
| Profiles | If the Default Group is set to 'User Profile' (The most recently selected group in the Group Menu is the default group displayed). Administration page / Additional settings |
| UserPics | If you display a picture for each user and the user picture definition is set in the database (UserPicDefInDB = True in calagent.inc) |
| **Meetings** | **When meetings are booked through the AgendaX web interface and meeting invitations are not sent ('Send invitation' is not checked in the 'Book meeting' form when sending a meeting invitation), AgendaX books meetings individually into attendees mailboxes. No relationship between those individual meetings exists on Exchange Server.** |

| | |
|---|---|
| | **AgendaX therefore keeps the relationship in the Meetings table for such meetings, so that they can later be modified or deleted.**<br>**Meetings are deleted from this table after the specified amount of time in the configuration has passed (AgendaXCfg.exe / Meeting / Booking reference in database)** |

All other tables **do not need to be backed up**, as they are constantly being rewritten by the AgendaX Update Service. All tables are recreated in case they are deleted. All tables are automatically created when AgendaX is configured to work with an empty database when the AgendaX Update Service is run for the first time.