


Reference guide SDI and e-mail from within Adlib

Adlib
an AXIELL product

Axiell ALM Netherlands BV

Contents

1 Sending e-mail from Adlwin

From Adlib 5.0, e-mails can be sent through an e-mail server from Adlwin (used by most of your Adlib applications).
So now you can send out reminders per e-mail directly from Adlwin, without the need for the standard Windows e-mail program.
For convenience, we will be using the following terms in the rest of this chapter:

- **MAPI**: the use of the standard Windows e-mail program
- **SMTP**: the direct use of a mail server.

1.1 Differences between MAPI and SMTP

If a program uses the Windows MAPI, e-mail is sent through the standard mail program of the currently logged-on user. This can be handy at times, but there are also cases in which it is only confusing or slow.
The most important problems with this are the following (with Microsoft Outlook as an example of MAPI, but this goes for any e-mail program):

- Microsoft Outlook does not allow you to use a different sender than the one currently logged on.
- Outlook will issue a standard warning that a program is trying to send e-mail through Outlook (a safety measure).

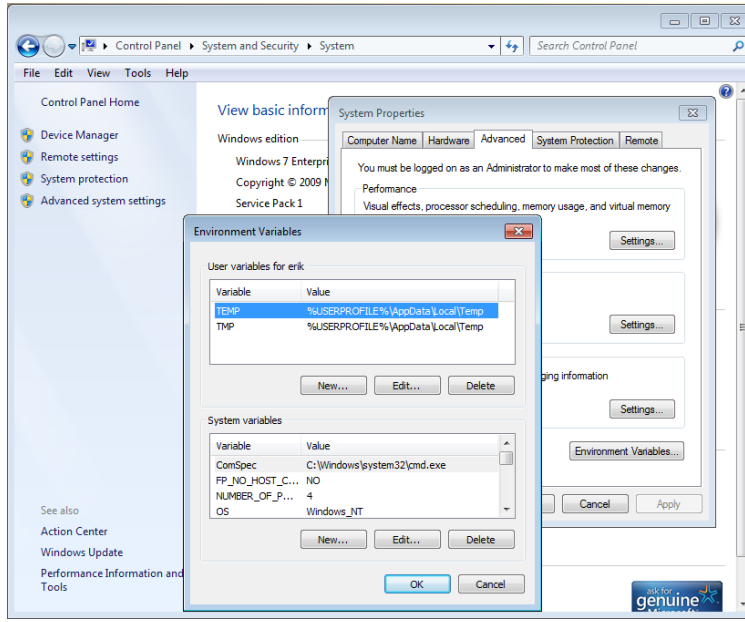
The first point is especially bothersome when a library for instance wants to send out reminders to borrowers from the registration office's general e-mail address instead of the employee's own e-mail address.
The second can cause problems when you are sending out 100 e-mails at the same time.
If a program such as adlwin.exe accesses a mail server directly through SMTP (Simple Mail Transport Protocol), these problems will not occur. This is because the program composes and sends the e-mails itself, without intervention from other programs. All users will then send e-mails from within Adlib via the same e-mail server. However, a disadvantage of SMTP is that an e-mail can only have one attachment; if an adapl tries to send an e-mail with multiple attachments anyway, an error 332 will be generated.

1.2 Instructions for installation

To let Adlwin know that it should send out e-mails directly through SMTP instead of through MAPI, two conditions must be met:

- The file EVALSMTP.DLL must be present in the directory that also contains the Adlib executables.
- The variable ADLIB_SMARTHOST must be set to the name of the mail server Adlwin should use for sending the e-mails through SMTP.
  This environment variable can be set in two ways:

  1. by entering the line, `setvar('ADLIB_SMARTHOST', 'name-of-my-smtp-server')` in the ADAPL file from which e-mails are sent, preferably at the start of the file (or at least before an e-mail is sent). You can only send out e-mails from within the adapl with this code!
  2. by assigning a value to the variable in your Windows system environment by opening *Control panel* > *System* via the *Start* button and subsequently opening the *Advanced* tab in the *System settings* window via the *Advanced system settings* option, where you click the *Environmental variables* button.
     Here you can create a new variable called `ADLIB_SMARTHOST`. You can use the address (or name) of the mail server (SMTP server) as its value. Whether the environmental value should be created as a *User variable* or a *System variable*, depends on whether you want all Adlib users to be able to send e-mails (system variable) or specific users only (make a user variable for each of them).



■ Consequences of choosing an installation

The choice you make when setting the environmental variable ADLIB_SMARTHOST, has consequences for the consistency and operation of your system. If you set the environment variable once in your Windows system, it applies to all users of that computer, but if the environment variable is set in an adapl then it only applies during the execution of that adapl and not outside of it.
If you choose to set the variable in your Windows system, you have to do this on every workstation from which e-mails can be sent (from Adlwin), in order to prevent differences when sending out e.g. reminders.
If the environmental value has not been set, or if the EVALSMTP.dll file (or one of the four Microsoft DLLs) is missing, then e-mails will still be sent through MAPI (unless the e-mail is sent from within an adapl and the environment variable has been set in the adapl).

■ An example

To clarify matters, the following is a small piece of ADAPL code:

```
setvar('ADLIB_SMARTHOST', 'saturnus.nl.adlibsoft.com')
gosub 300  /* send e-mails
…
300
```

```
      /* sendmail(sender e-mail address, mail-to-address,
      cc-address, e-mail subject, text, attachments)
sendmail('info@library.uk', 'borrower_johnsen@home.uk', '',
'Reminder', contents, '')
return
```

In this example, the text of the e-mail is contained in the variable `contents`.

1.3 Setting a global SMTP sender e-mail address

When you send e-mail from within Adlib, either MAPI or SMTP will be used. Usually the (ADAPL) output format determines how and when e-mail is going to be sent and the sender e-mail address and the SMTP server name to be used are taken from *adlib#.txt* files. If you would like the sender e-mail address to be a general address of your organisation, then begin by setting it in those files (in lines 192 and 278 actually).

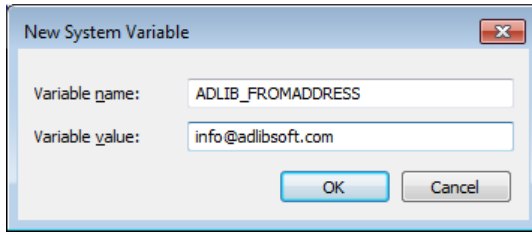
However, there are circumstances in which ad hoc e-mail must be sent from within Adlib, through SMTP. After stepping through the *Print wizard* for example, the Windows *Print* window usually opens, allowing you to decide to send output originally meant to be printed via e-mail after all, and select some addressees at the spot. If Adlib cannot use MAPI at that point, as can be the case on hosting servers for instance, then so-called environment variables of the operating system must have been set to instruct Adlib at least which SMTP server to use. This can done with the `ADLIB_SMARTHOST` variable containing the address or name of the SMTP mail server to use. In this case the login name of the current user will be used as the sender in the e-mails. However, because a name is not a valid sender or reply e-mail address, mail servers will ofter block such e-mail. Therefore a second environment variable is available: `ADLIB_FROMADDRESS` in which an e-mail address must be stored as the value. This e-mail address wil then be used as the general sender e-mail address when ad hoc e-mail is being sent via SMTP.

This enviroment variable can be set, just like ADLIB_SMARTHOST, as follows:

1. Via the Windows *Start* menu, open your *Control panel*, choose *System and security* > *System* and click *Advanced system settings* in the vertical bar on the left.
   Then click the *Environment variables* button in the *Advanced* screen tab of the *System properties* window to open the *Environment variables* window.

2. In the upper half of the *Environment variables* window you can set user variables for yourself – these are only valid if you are logged on – and/or system variables which apply to everyone using this system. If the computer (client) is part of a local network, then both settings apply to this computer only. If you make these settings on a hosting or network server, then they apply to users logged in on that server.

3. Click the *New* button under *User variables* or *System variables*, for *Variable name* fill in `ADLIB_FROMADDRESS` and for *Variable value* fill in the full sender e-mail address. Close all windows by clicking *OK*.



2 SDI

SDI stands for Selective Dissemination of Information. This means that you can keep your customers (for instance library visitors or literature researchers) abreast with information that is relevant to them. In practice, it means that you can save interest profiles. Based on these profiles, the Adlib system is automatically searched for information, which is then presented nicely. You could use this to send your customers regular e-mails about new additions that are relevant to them.
The way we have implemented this functionality makes sure that it can also be used for other things. For example, it is possible to automatically generate reminders in the loans module, without having to press any buttons, and to send them out as e-mails. You can also automatically send out the most recent list of additions, or just an overview of additions pertaining to a particular subject, as can be indicated per customer. SDI is also very suitable for automatically reminding customers of reservations that are ready to be picked up, or about the expiry of a loans contract.

The SDI functionality has been constructed as follows: in your Adlib application (running on *adlwin.exe*) or an Adlib Internet Server web application (running on *wwwopac.ashx/.exe*), a user creates an interest profile (really just a pointer file based on a search statement, with extra properties specific to SDI). An existing interest profile can also be edited via the same applications.
Further, an SDI server needs to be created, based on *sdi.exe* (or *wwwopac.exe* which offers the same functionality). Whenever that program is being run it will execute the search statement stored in the interest profile again if the frequency settings allow it to, it will update the pointer file and will print or e-mail the search result: the Adlib application and the Internet Server themselves are not involved in that process.
Although you can run *sdi.exe* manually from the server via a batch file or (DOS) command line, it is much more convenient to use the Windows Task Scheduler on that server to regularly run the program via a batch file automatically so you won't have to bother anymore.
(*Sdi.exe* does not run continuously in the background: after it has processed the relevant interest profiles, it closes automatically.)

2.1 Creating a search profile

In Adlib, an interest profile consists of two parts:

- the search query, stored in a pointer file with SDI settings amongst which the destination of the output (e.g. e-mail);
- layout instructions.

If you put together a profile yourself, use your Adlib application for this. As pointer files often already contain search queries, SDI has been integrated with pointer files.

1. Use the *Advanced search* to enter an explicit search statement, as you would usually do.
2. Turn the search query into a pointer file with the *Write set* button. Assign a title to indicate what the search query is about.
3. Open the *Pointer files* window with the button in the Expert Search System or with the button in the *Start* menu (when a data source has been selected already).
4. Right-click the pointer file of which you want to create, edit or view an SDI profile, and choose the *Scheduled searches* option in the pop-up menu which appears. On the *SDI General, SDI Destinations* and *SDI Schedule* tabs in the *Scheduled searches* window you can set or view the SDI profile for the current pointer file.
5. Specify the profile via de available properties (see below) and click *OK*. *Sdi.exe* on your server (executed automatically by Windows Task Scheduler) will now regularly and automatically process this profile. You do not need to start an Adlib application first.

An SDI profile has the following properties:

1. **Language**. Select the language in which the search statement for the pointer file has been put together. For example: if you used English field names in the search statement, then you should set the language here to English as well, otherwise Adlib cannot find the fields later on.
2. **Format**. Enter the path to an adapl (without the .bin extension) or stylesheet (.xslt) to provide the layout for the output. The path must be relative to the folder which holds sdi.exe. For example: if sdi.exe is located in the Adlib *executables* subfolder and a stylesheet is located in the Adlib *museum* subfolder, then the path would be something comparable to *..\museum\mystylesheet.xslt*. If the adapl or stylesheet is located

in the same folder as sdi.exe, then you only need to provide the file name, without path. The *Format* option is mandatory.

3. **Pruning**. The search result can be filtered before it is sent out. Choose *New records* to only show records that have not been reported earlier, or report only new or modified records by selecting *New or changed records*. The option *DM/DI changed* does almost the same, but is application-dependant. You can also report the full search result by selecting *No pruning*. The *Undefined* option has no meaning, you have to choose one of the options below it.
4. **Limit**. Specify the maximum number of records the search result may contain. 0 means that there is no limit.
5. **Subject**. This field can be used to add a subject line to an e-mail.
6. **Comments**. Note possible comments about this SDI profile.
7. **Mode**. The search result can be sent to two different output systems. Select *Email* or *Printer*. Both when you print or e-mail, all records will be placed underneath each other. The adapl can make use of the normal reserved tags to print headers, footers and record separators.
8. **Printer destination**. Fill in nothing to use the default printer. If you want to use another, then provide the full path to it here.
9. **E-mail format.** Choose *HTML* if you've transformed the search result to an HTML page (by means of an XSLT stylesheet for example). The e-mail will then show the browser display of that page instead of the HTML code itself. In all other cases, choose *Plain text*.
10. **Email addresses**. Provide all e-mail addresses to which the result of the current search query has to be sent regularly. Type an e-mail address in the left entry field en move it to the list on the right by clicking the > button. Repeat this for all e-mail addresses. You can remove an address from the list by selecting it and clicking the < button.
11. **Frequency and Schedule**. Times and dates when the user wants his or her search profile to be executed. First choose a frequency, then fill in more specific data on the right in the entry fields of Schedule that become active (which ones become active depends on the set frequency). This way you can set the *Hour* for *Daily*, and also the day for *Weekly* and *Monthly*, and for *Yearly*: *Hour*, *Day* and *Month*.

Remember, however, that the frequency can be limited by the Windows Task Scheduler settings for the SDI program on the

server. If the network administator has set the program to update only once a day, you will not be able to receive a new update more often than that. When the SDI server is actually running, it checks whether the SDI profiel must be processed by determining if the frequency setting in that profile and the date of the last update stored in there as well means that it should have been done already: so the frequency settings do not need to match the Task Scheduler settings (perfectly).

12. **Expiry date**. Select a date after which you no longer want to execute the search profile.
13. **Suspended**. As long as this option is marked, the profile will not be executed.
14. **Last run**. The date on which a search profile was executed last. You cannot add or modify this date yourself.

2.2 Programming a layout adapl

An SDI adapl is only programmed and used to lay out an SDI search result. You can program the adapl like you would program a print adapl for plain text, using PRINT and OUTPUT statements and such. You can use existing Adlib output formats (for plain text) from the model application or use them as examples. The SDI program itself sends out the e-mails or makes the printouts. This adapl is called for every record in the resulting set. (Save the compiled adapl in the directory that contains the SDI server.)
Some familiar reserved ADAPL variables suitable for executing an SDI search profile are:

Tag	Type	Use for
&I and &B[1..3]	numeric	Serial number of record in the list
&6[1..3]	text	Database data
&P	numeric	Number of the active language
&1	numeric	Contains the value 25 (new execution code for SDI)
&L, &T and others		

The following (optional) extra reserved variables are available for SDI:

Tag	Type	Contains
&Q[1]	text	subject, for e-mail
&Q[2]	text	comments
&Q[3]	text	Suspended
&Q[4]	text	Owner (maker of this profile)

&Q[5]	text	Search query
&Q[6]	text	Sort command
&Q[7]	text	Frequency, as meant for SDI (see next paragraph)
&R[1]	numeric	Pruning. Values: 1 (do not show previously reported records), 2 (show records in which the date of entry DI or the date of modification DM has been changed), 3 (no restriction), or 4 (only show new or modified records). DM and DI are registered depending on the application.
&R[2]	numeric	Delivery mode. Values 1 (e-mail) or 2 (printer)
&U[1]	text	Creation date of this search profile
&U[2]	text	Creation time of this search profile
&U[3]	text	Previous execution date of this search profile
&U[4]	text	Previous execution time of this search profile

See the Designer Help for more information about the ADAPL programming language.

■ **Frequency**

You set the frequency with which the search profile ideally should be executed mostly interactively in the Adlib application or in Internet Server, but in the background this is stored in a special format, which you have to be familiar with if you use the Q[7] variable in an sdi adapl or when you do development work on your Adlib Internet Server application.

Remember, however, that the frequency can be limited by the Windows Task Scheduler settings for the SDI program on the server. If the network administrator has set the program to update only once a day, you will not be able to receive a new update more often than that.

A frequency consists of five (strings of) values in a fixed order, separated by spaces, to indicate dates and times on which the search profile must be executed. A search profile will only be executed when the specified time and month are current, and at least the day of the month or the week concur with the current date. Possible values are:

Partial parameter	**Order**	**Valid values**
minutes	1	0 through 59, or *
hour	2	0 through 23, or *
day of the month	3	1 through 31, or *

month	4	1 through 12, abbreviated names, or *
day of the week	5	0 through 7 (in which 0 and 7 represent Sunday), abbreviated names, or *

The allowed values can be used as follows:

- An asterisk (*) means that you wish to use any value within the range, for instance every day of the week.
- Abbreviated names for days are: sun, mon, tue, wed, thu, fri, sat. For the months these are: jan, feb, mar, apr, may, jun, jul, aug, sep, oct, nov, dec. (These are not case-sensitive.)
- Ranges or lists of values can also be specified for one parameter. You specify a range by placing a dash between two numbers. So if you specify 8-11, that means the search profile is executed at 8, 9 10 and 11 o'clock.
- For a range or an asterisk, you can also specify an interval. Follow the range with a slash (/) and an interval number. If for instance you want the profile executed every two hours, specify the hour parameter in two ways: 0-23/2 or */2.
- A list may consist of several numbers and/or ranges, separated by commas, e.g.: 0-4,6,8-12. (Do not use spaces after the commas.)
- A range or a list of abbreviated names is not allowed.
- If both the day of the month and the day of the week are designated by numbers, the search profile is executed on all specified days. In other cases, the average of both collections will be used (see for instance the definition of @yearly).
- Instead of these five parameters you can also use one of the following special strings for setting a simple frequency when the time of the search is not very relevant:

String	**Description**	■ **Parameter equivalent**
@yearly	once a year	0 0 1 1 *
@annually	same as @yearly	
@monthly	once a month	0 0 1 * *
@weekly	once a week	0 0 * * 0
@daily	once a day	0 0 * * *
@midnight	same as @daily	
@hourly	once an hour	0 * * * *

More examples:

- **5 0 * * *** Execute the search profile every day, five minutes after midnight.
- **15 14 1 * *** Execute the search profile on the first day of every month at 2:15 PM.
- **0 22 * * 1-5** Execute the search profile every working day at 10 PM.
- **23 0-23/2 * * *** Execute the search profile every day at 0:23 AM, and then every two hours.
- **5 4 * * sun** Execute the search profile every Sunday at 4:05 AM.

2.3 Programming an XSLT layout stylesheet

Instead of an adapl you can use an XSLT stylesheet to apply a layout to the search result of an executed SDI profile. Under the hood of Adlib, all data is processed as XML. With an XSLT stylesheet the XML can be transformed to HTML, for example, while HTML can be displayed in SDI e-mails.
So you have to create an XSLT stylesheet to transform Adlib XML of records from a certain database into HTML. Assume this is grouped XML as generated by the Adlib API too. Use the example stylesheet below for museum objects or use one of the Adlib Office Connect stylesheets to start with. You are of course free to further adjust these stylesheets to your requirements. Place the stylesheet in a suitable location in your Adlib Software folder, maybe the folder with the name of the application, or in a new *\stylesheets* or *\xslt* subfolder.

As an illustration, the example code also contains a template for the *reproduction.reference* field, which makes sure that a link to the images belonging to a record are included in the output, so that those images become visible in the printout or e-mails as well. This template assumes that the field only contains a file name (without path) and that the images folder in this example can be found on *\\server1\Adlib\images*. You should change this UNC path to the UNC path to the folder which actually contains your images. The images themselves are not included in the e-mail – an e-mail could easily become to large – only a link to it. This means that this template has limited applicability for e-mails, namely only for e-mails sent within your local network in which everyone has access to this UNC path. SDI e-mails opened on computers outside the network, would not show any images.
If you have an Adlib images server, you could use the wwwopac calls (URLs) for retrieving images to include links to images in e-mails

which should also work outside the local network.
You can also leave out the processing of images entirely of course, by simply not including a template for it in the stylesheet.

Further note that if SDI doesn't send any e-mails while you did set it up like that, one of the possibilities is that something is wrong with your XSLT stylesheet. E-mails can only be sent when your stylesheet is a 100% okay syntactically.

```
<?xml version="1.0" encoding="utf-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">

  <xsl:output method="html" encoding="UTF-8" indent="yes"/>

  <xsl:template match="/adlibXML">
   <html>
   <head>
     <STYLE type="text/css">
       p {font-family:"verdana"; font-size: 10pt}
     </STYLE>
   </head>
   <body>
        <xsl:apply-templates select="recordList"/>
   </body>
   </html>
  </xsl:template>

  <xsl:template match="recordList">
    <xsl:apply-templates select="record"/>
  </xsl:template>

  <xsl:template match="record">
      <p>
        <xsl:apply-templates select="priref"/>
        <xsl:apply-templates select="object_number"/>
      </p>
      <xsl:if test="count(Object_name) != 0">
      <p>
       <table border="1" cellpadding="10"
       style="border-collapse: collapse" width="100%">
        <tr>
          <td width="100%" bgcolor="#EAEAFF">
           <p>
             <xsl:apply-templates select="Object_name"/>
             <xsl:apply-templates select="other_name"/>
           </p>
           <p>
              <xsl:apply-templates select="Description"/>
           </p>
           <xsl:if test="Production/creator != ''">
            <p>
              <xsl:apply-templates select="Production"/>
            </p>
           </xsl:if>
```

```
          </td>
         </tr>
        </table>
      </p>
    </xsl:if>

    <p>
      <table border="1" cellpadding="10"
       style="border-collapse: collapse" width="100%">
       <tr>
        <td width="100%" bgcolor="#F9F9F9">
         <p>
          <xsl:if test="physical_description != '' or
           Dimension/dimension.value != ''">
             <xsl:text>Physical description:</xsl:text>
             <br/><br/>
          </xsl:if>
          <xsl:apply-templates select="physical_description"/>
          <xsl:if test="Material/material != '' or
           Dimension/dimension.value != ''">
           <p>
           <xsl:if test="Material/material != ''">
             <xsl:apply-templates select="Material"/>
           </xsl:if>
           <xsl:if test="Dimension/dimension.value != ''">
             <xsl:if test="Material/material != ''">
               <xsl:text>, </xsl:text>
             </xsl:if>
             <xsl:apply-templates select="Dimension"/>
           </xsl:if>
           <xsl:text>.</xsl:text>
           </p>
          </xsl:if>
          <xsl:if test="Reproduction/reproduction.reference != ''">
          <p>
             <xsl:apply-templates
              select="Reproduction/reproduction.reference"/>
          </p>
          </xsl:if>
       </p>
       </td>
      </tr>
    </table>
  </p>
  </xsl:template>

  <xsl:template match="reproduction.reference">
    <p>
     <img src="\\server1\Adlib\images\{normalize-space(.)}" />
    </p>
  </xsl:template>

  <xsl:template match="priref">
    <xsl:if test="position() = 1">
       <xsl:text>Record: </xsl:text>
```

```
      <xsl:value-of select="."/>
       <br/>
    </xsl:if>
 </xsl:template>

  <xsl:template match="object_number">
      <xsl:text>Object: </xsl:text>
      <xsl:value-of select="."/>
      <xsl:apply-templates select="../object_category"/>
  </xsl:template>

  <xsl:template match="object_category">
    <xsl:text>  (</xsl:text>
    <xsl:value-of select="normalize-space(.)"/>
    <xsl:text>)</xsl:text>
  </xsl:template>

  <xsl:template match="Object_name">
    <strong>
      <xsl:if test="position() &gt; 1">
        <xsl:text> &amp; </xsl:text>
      </xsl:if>
      <xsl:value-of select="object_name"/>
    </strong>
  </xsl:template>

  <xsl:template match="other_name">
    <xsl:text>, other name: </xsl:text>
    <xsl:value-of select="."/>
  </xsl:template>

  <xsl:template match="physical_description">
       <xsl:value-of select="."/>
  </xsl:template>

  <xsl:template match="Description">
    <xsl:if test="description != ''">
          <xsl:value-of select="description"/>
    </xsl:if>
  </xsl:template>

  <xsl:template match="Material">
    <xsl:choose>
      <xsl:when test="position() = 1">
           <xsl:text>Made of </xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text> &amp; </xsl:text>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:value-of select="material"/>
  </xsl:template>

  <xsl:template match="Dimension">
    <xsl:if test="position() &gt; 1">
```

```
      <xsl:text>, </xsl:text>
    </xsl:if>
    <xsl:value-of select="dimension.type"/>
    <xsl:if test="dimension.part != ''">
      <xsl:text> (</xsl:text>
      <xsl:value-of select="dimension.part"/>
      <xsl:text>)</xsl:text>
    </xsl:if>
    <xsl:text> </xsl:text>
    <xsl:value-of select="dimension.value"/>
    <xsl:text> </xsl:text>
    <xsl:value-of select="dimension.unit"/>
  </xsl:template>

  <xsl:template match="Production">
    <xsl:variable name="pos" select="position()"/>
    <xsl:choose>
      <xsl:when test="$pos = 1">
         <xsl:text>Created by </xsl:text>
       </xsl:when>
      <xsl:otherwise>
        <xsl:text> &amp;amp; </xsl:text>
      </xsl:otherwise>
    </xsl:choose>
    <xsl:apply-templates select="creator"/>
    <xsl:if test="position() = last()">
       <xsl:apply-templates select="../Production_date[1]"/>
       </xsl:if>
  </xsl:template>

    <xsl:template match="Production_date">
    <xsl:text> </xsl:text>
    <xsl:choose>
      <xsl:when test="(production.date.start !=
       production.date.end) and (production.date.end != '')">
            <xsl:text>between </xsl:text>
           <xsl:value-of select="production.date.start"/>
            <xsl:text> and </xsl:text>
           <xsl:value-of select="production.date.end"/>
           <xsl:text>.</xsl:text>
      </xsl:when>
      <xsl:otherwise>
        <xsl:text> in </xsl:text>
        <xsl:value-of select="production.date.start"/>
        <xsl:text>.</xsl:text>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

  <xsl:template match="creator">
    <xsl:choose>
      <xsl:when test="contains(., ',')">
        <xsl:value-of select="substring-after(. , ',')"/>
        <xsl:text> </xsl:text>
        <xsl:value-of select="substring-before(. , ',')"/>
```

```
      </xsl:when>
      <xsl:otherwise>
        <xsl:value-of select="."/>
      </xsl:otherwise>
    </xsl:choose>
  </xsl:template>

</xsl:stylesheet>
```

2.4 Client-commands

To communicate with the SDI server from the SDI client-side, you could use a URL or a batch file on the server containing an command line to execute *sdi.exe* or *wwwopac.exe* with certain parameters. Besides the standard *wwwopac.exe/sdi.exe* parameters to indicate the desired database and to specify a search query (see the examples below), you can add the `SDICMD` parameter (amongst others) to the CGI string (the remaining part of the requesting URL string immediately after the "?") or to the arguments string in a command line, to address the SDI functionality; the following commands can be assigned to `SDICMD`: (When you use these settings, leave off the brackets < and >, and any single quotation marks around values.)

- `STORE`. With `SDICMD=store` you can create and save an SDI profile. In the CGI/arguments string, first specify the target database and the search query that you'd like to save in the profile: if needed, see the WWWOPAC Reference guide for information about the syntax of search queries. Further keep in mind that certain reserved characters are not allowed in the CGI string, while they *are* allowed to appear in a command line.
  The following parameters are available for defining the SDI profile:
  - `SDIFREQ=<frequency>` See paragraph 2.2.
  - `SDIEMAIL=<e-mail address>` Use this parameter when you want to send e-mail to only one person; because in an existing e-mail addresses list the first field will be overwritten. That's why you should use `SETEMAIL` for making a list.
  - `SDIADAPL=<lay-out adapl>`, mandatory if you are not using `SDIXSLT`.
  - `SDIEXPIRES=<expiry date>`
  - `SDISUSPEND=<'true', '1', or 'on': suspended>`
  - `SDIPRUNEMODE=<'1', '2', '3', of '4': pruning>`

- SDIDELIVERYMODE=<'1' or '2': delivery mode, respectively e-mail or printer>
- SDIMAILFORMAT=<0|1> This parameter can be used to specify the e-mail format (other than the *Plain text* format which was the default format before 6.5.2), and store this in the pointer file. 1 means *HTML*, 0 means *Plain text*.
- SDISUBJECT=<e-mail subject>
- SDIPDEST=<printer destination for printing> Also set SDIDELIVERYMODE to 2.
- SDIXSLT=<stylesheet for layouting instead of adapl>, mandatory if you are not using SDIADAPL. The name should end with .xsl or .xslt, and you can't set a font or page size with this.
- DATABASE=<name of the target database as set in the configuration file for the SDI server>

For more information about what to enter, see the comparable information about the same options earlier in this text.

An example of using a wwwopac.exe URL to create an interest profile:

```
http://mysite.com/wwwopac/wwwopac.exe?database=
Collect&search=IN=1995*&SDICMD=store&SDIEMAIL=erik@
hotmail.com&SDIADAPL=docfull
```

An example of using an sdi.exe URL to create an interest profile:

```
http://mysite.com/wwwopac/sdi.exe?database=Collect&
search=IN=1995*&SDICMD=store&SDIEMAIL=erik@hotmail.
com&SDIADAPL=docfull
```

An example of using a command line/batch file to create an interest profile:

```
sdi.exe "database=Collect&search=IN=1995*&SDICMD=
store&SDIEMAIL=erik@hotmail.com&SDIADAPL=docfull"
```

Note that it is important that the databasepath setting in the configuration file for the SDI server (see the next chapter) on network servers contains a UNC path, no drive denotations. It is also required that you start such a batch file from the server itself, via a remote desktop connection if you are currently not logged in on that server. (By the way, a batch file is nothing more than a text file with the *.bat* extension.)

- FIND. With SDICMD=find you can search for a specific SDI profile and its number and title, and retrieve it through the e-mail address of a visitor. Use the following parameters:
    - SDIKEY=<'email', 'number' or 'title'>
    - SDIVALUE=<e-mail address, number or name, depending on SDIKEY> E-mail addresses use 'contains' to search for the value indicated here; searching for '@' will yield all SDI profiles with e-mail addresses.
    - DATABASE=<name of the target database as set in the configuration file for the SDI server>

    An example of using an sdi.exe URL to find an interest profile:
    http://mysite.com/wwwopac/sdi.exe?database=Collect&SDICMD=find&SDIKEY=email&SDIVALUE=erik@hotmail.com
- REMOVE. With SDICMD=remove you can delete an SDI profile. You will need the number of the SDI profile (SDI pointer file) to do this:
    - SDIVALUE=<SDI profile number>
    - DATABASE=<name of the target database as set in the configuration file for the SDI server>

    An example of using an sdi.exe URL to delete an interest profile:
    http://mysite.com/wwwopac/sdi.exe?database=Collect&SDICMD=remove&SDIVALUE=56
- UPDATE. With SDICMD=update you can change an SDI profile you created earlier. As desired, you will need one or more arguments from STORE (see above), as well as the number of the profile:
    - SDIVALUE=<SDI profile number>
    - DATABASE=<name of the target database as set in the configuration file for the SDI server>

    An example of using an sdi.exe URL to edit an existing interest profile:
    http://mysite.com/wwwopac/sdi.exe?database=Collect&SDICMD=update&SDIVALUE=57&SDISUBJECT=recent%20acquisitions
- RUN. With SDICMD=run you execute all search profiles for a single database or for all databases:
    - SDIKEY=<'1': to execute all search profiles for all databases, irrespective of *Suspended*, *Expires* or *Last*> Useful for test purposes.

- `DATABASE=<name of the target database as set in the configuration file for the SDI server>` Only specify this if you want to execute the SDI profiles for one database only. If this database alias hasn't been specified in the configuration, then either all databases will be processed or the log will report an error message.

An example of using an sdi.exe URL to process all existing interest profiles in the Collect database, except profiles which have been suspended, expired or just not up yet (considering their frequency):

```
http://mysite.com/wwwopac/sdi.exe?database=Collect&
SDICMD=run
```

An example of using a command line/batch file to process all existing interest profiles in all database, except profiles which have been suspended, expired or just not up yet (considering their frequency):

```
sdi.exe "SDICMD=run"
```

Note that it is important that the `databasepath` setting in the configuration file for the SDI server (see the next chapter) on network servers contains a UNC path, no drive denotations. It is also required that you start such a batch file from the server itself, via a remote desktop connection if you are currently not logged in on that server. (By the way, a batch file is nothing more than a text file with the *.bat* extension.)

- `DELEMAIL`. With `SDICMD=delemail` you remove one or all e-mail addresses from an SDI profile:
  - `SDIKEY=<index number or '-2'>` Filling in `-2` removes all e-mail addresses from this profile, filling in a specific index number removes only that address from the list; index number 0 is the first e-mail address, 1 is the second, etc.
  - `SDIVALUE=<SDI profile number>`
  - `DATABASE=<name of the target database as set in the configuration file for the SDI server>`
- `SETEMAIL`. With `SDICMD=setemail` you add an e-mail address to an SDI profile:
  - `SDIKEY=<index number or '-1'>` Filling in `-1` adds the new address to the back of the list automatically, filling in a specific index number inserts the address in that place on the list and therefore overwrites the address that was

already there; index number 0 is the first e-mail address, 1 is the second, etc.

- `SDIVALUE=<SDI profile number>`
- `DATABASE=<name of the target database as set in the configuration file for the SDI server>`
- `SDIEMAIL=<e-mail address to be added>`

2.5 Configuration of SDI

A typical configuration of SDI, regardless of whether you have an Internet Server web application or not, is the following:

■ Required files

Make sure that you have an Adlib *\executables* or *\wwwopac* folder on the server which will have to function as the SDI server. This can be an existing folder of your Adlib application or Internet Server system or a copy thereof, and the folder name may be changed if you desire. The point is that the folder contains *sdi.exe* and/or *wwwopac.exe*, amongst others: as far as SDI functionality is concerned, those programs are interchangeable. The folder also needs to contain an SDI configuration file (see next paragraph), an Adlib license file (*adlib.lic*) and the following DLLs: *adliblic.dll, adlibU.dll, adlibweb.dll* and *evalsmtp.dll*.

■ The SDI configuration file

For the proper functioning of SDI you'll have to make a few settings in a configuration file. You can put those settings in a text file to be created under the name *default.sdi* or in an *adlibweb.xml* file, in the same folder as *sdi.exe* or *wwwopac.exe*. If you need specified database aliasses to be able to process interest profiles per database, then you'll have to put the extra settings in (a copy of) your Internet Server *adlibweb.xml* file or in a separate *adlibweb.xml* file (if you do not have Internet Server) in the same folder. (The problem is that *default.sdi* up till now does not accept a database specification. Other than that, the settings are interchangeable.) We recommend to use *adlibweb.xml* anyway, since *default.sdi* isn't always recognized.

The following variables and settings are specific to SDI:

- ADLIB_SMARTHOST=<your SMTP host for sending SDI mailings: this setting is mandatory>
- SDIFROM=<e-mail address from which the mailing is sent: this setting is mandatory>
- DM=<Adlib tag or field in which the modification date of a record is saved. This has to be an indexed field. The DM-setting is optional and only relevant if the interest profile option *Pruning* ever gets set to *DM/DI changed*.>
- DI=<Adlib tag or field in which the entry date of a record is saved. This has to be an indexed field. The DI-setting is optional and only relevant if the interest profile option *Pruning* ever gets set to *DM/DI changed*.>

An example of an *adlibweb.xml* file (only the part relevant to SDI is given):

```
<?xml version="1.0" encoding="UTF-8" ?>
<webConfiguration>
  <globalConfiguration>
    <databasepath>\\ourserver\adlib\application 4.2
                  SQL\data</databasepath>
    <xmltype>grouped</xmltype>
    <sdifrom>erik@mymuseum.com</sdifrom>
    <adlib_smarthost>smtp.mymuseum.com</adlib_smarthost>
    <dm>dm</dm>
    <di>di</di>
  </globalConfiguration>

  <databaseConfiguration database="Collect">
    <database>collect&gt;intern</database>
  </databaseConfiguration>

</webConfiguration>
```

The same example in the shape of *default.sdi* (without specified database though):

```
ADLIB_SMARTHOST=smtp.mymuseum.com
DATABASEPATH=\\ourserver\adlib\application 4.2 SQL\data
SDIFROM=erik@mymuseum.com
DM=dm
DI=di
```

■ Settings in the Windows Task Scheduler

In the Windows Task Scheduler you must make some settings if you'd like your batch file* running *sdi.exe* or *wwwopac.exe* to be executed automatically on a regular basis. Of course it depends on the version of Windows in exactly what way you can make these settings. See Microsoft's documentation for more information about that.

Under Windows Server 2003 R2 this can be done as follows:

1. In Windows, click *Start* > *All programs* > *Accessories* > *System tools* > *Scheduled tasks*.
2. Double-click the *Add scheduled task* option in the right window pane.
3. A wizard will guide you through the settings for a regular execution of your batch file.

Every time profiles are being executed, a *lastrun_sdi.dat* file (4 bytes) will be saved in the folder in which the *adlibweb.xml* or *default.sdi* file and the SDI server are located: this *.dat* file contains the date and time at which the Task Scheduler last executed the SDI search profiles.

* You can create a batch file to run *sdi.exe* or *wwwopac.exe* with parameters when the Task Scheduler starts the file or when you double-click it yourself. Simply create a text file with the *.bat* extension, name it `sdi.bat` for instance, place it next to *sdi.exe* or *wwwopac*.exe and put a command line like the following in there:

```
sdi "SDICMD=run" > sdilog.xml
```

This will execute all applicable SDI profiles after which an XML log file (here *sdilog.xml*) will be created in the same folder, containing the result of the action (which is convenient if the batch file is executed by the Windows Task Scheduler on the server).
To test your SDI server before you have the Task Scheduler execute the SDI profiles on a regular basis, you can double-click the saved batch file to execute it this once.

2.6 Requirements for sdi.exe

On the server computer you will need the following programs to use *sdi.exe*:

- minimally Windows Vista.
- an Adlib application (with a subfolder *data*).

- the sdi executable, plus accompanying files, amongst which *default.sdi*, *adlib.lic* and *adlibU.dll.* For the complete listing see chapter 2.5.
- MSXML4 or higher. Version 4.0 SP2 of the MSXML parser (or a newer version) from Microsoft should be installed. You can check this in your registry (*Start* > *Execute*, type `regedit` and click *OK*). In it, search for the text `MSXML`. If the parser is not present, then you can download this software from the Microsoft web site (http://www.microsoft.com); from their homepage, search for `MSXML4` or `MSXML 6.0`. Click e.g. the hyperlink *Download Microsoft XML Core Services (MSXML) 4.0 Service Pack 1*. Then choose the *msxml.msi* file and store it on your hard disk. Now install the software by double-clicking the file.