INTOGRATE AX – Developer

Lotus Notes Integration kit

Version 2.2

0. Introduction

0.1 AX to Lotus Notes integration kit

INTOGRATE AX – Developer enables thight integration between Dynamics AX and Lotus Notes.

The technology applied in this Integration Kit is based on integration to Lotus Notes via a Windows DLL program library. All administration and set-up are via AX.

The integration kit features

- **Possibility for on-line integration**
  All updates in e.g. Lotus Notes can be coordinated with updates in AX. It is possible to cancel updates, e.g. in the case of missing link.
- **Strict observance of security routines in both systems**
  It is not possible to evade system security routines, neither in Lotus Notes, nor in AX.
- **Embedding in standard AX code**
  Coding is done directly in the AX development environment via AX methods. This ensures that AX rules of integrity are kept.
- **Transparent access to Lotus Notes**
  Lotus Notes can be immediately accessed without the need for coding or changes in Lotus Notes design-elements. This e.g. enables integration with Lotus Notes applications for which the user has no access to source code but only knows the names of databases, forms and fields in Notes.
- **Sending of e-mails from Dynamics AX**
  Possibility for integration with E-mail system, by which e.g. work-flow systems can be implemented in AX.
- **Direct access to Lotus Notes API**
  Because the Integration Kit has been developed especially for Lotus Notes, the kit can deliver many Lotus Notes specific information and possibilities.

The integration kit is also available for the applications ATTAIN, CONCORDE XAL and CONCORDE C5.

0.2 Target group

As the integration tool contains a collection of tools for integration with Lotus Notes via AX's development environment, this manual is primarily intended for personnel with some knowledge of AX's programming facilities.

However, the level of programming knowledge required much depends on the degree of integration wanted between AX and Lotus Notes.

In order to use the integration kit, it is an advantage to have some knowledge of Lotus Notes, especially as regards the names of

- Databases
- Views
- Forms
- Fields

This information is required for in-house programming of integration solutions. However, skills in using the tools capable of supplying this information from Lotus Notes can be acquired in very short time. It is recommended that data transfer and manipulation in Lotus Notes be approved by the person(s) responsible for the company's Lotus Notes application(s).

0.3 Components

The integration kit consists primarily of two main components: an AX Class called "ax2ln", and a Windows program library called "ERP2LN.DLL".

0.3.1 Windows program library ERP2LN.DLL

As mentioned above, AX has limited facilities for direct communication with Lotus Notes. So the actual integration is not carried out within AX. This happens via a Windows program library, a so-called DLL file (Dynamic Link Library), which is one of the essential buildings blocks in Microsoft Windows. One of the purposes of these files is to create a common reference for the various Windows applications.

ERP2LN.DLL is what is termed an "API" by programming specialists (Application Programming Interface), i.e. program-based entry to a program.
With ERP2LN.DLL the AX programmer gets access to a number of Notes facilities which can be utilized outside Lotus Notes.
In principle, ERP2LN.DLL functions as if the workstation were linked to Lotus Notes as an ordinary user.

The integration kit is also available as a COM-component.

0.3.2 AX class "ax2ln"

The principal idea behind the integration kit is that all integration between AX and Lotus Notes should be carried out from inside AX.

The AX standard versions have very limited facilities for direct communication with Lotus Notes, it has been necessary to extend the functionality of AX. All extensions are contained in the AX class ax2ln. The primary purpose of this class is to offer an interface in the Notes integration that is as similar as possible to the options offered by AX.

NOTE: The DLL library should always be accessed via the ax2ln class and NOT accessed directly.

0.4 Training

A high level of training is not required in order to use the integration kit. A course will typically comprise the following areas:

1. The principles applied in integration between AX and Lotus Notes.
2. Integration kit commands.
3. Relevant Lotus Notes tools.
4. Integration case by which an AX application element is integrated with a Lotus Notes database.

The duration of the course depends on the participants' basic knowledge of Lotus Notes and the AX development environment, and on the level of detail by which the integration case is to be dealt with.

1. Program installation

1.1 Before you begin

1.1.1 System requirements

The following software is required:

- Microsoft Windows Workstation/Server 98, NT, 2000, XP, Vista, 7 or 8
- Lotus Notes client version 5.02c or later.[1]
- AX version 2.0 or newer.

1.1.2 Check list

Before you begin the actual installation of the Integration Kit it is recommended that you carry out the following points. They may prevent potential problems not related to the Integration Kit itself.

- Start your local Lotus Notes client. Check that you have access to the database you want to integrate to.
- Start AX and make sure that you have a valid user name and password for AX. Remember to exit AX before you continue with the installation.
- If you are using an AX client located on a common network drive, you must make sure that you have sufficient rights to modify the bin directory (e.g. "C:\Program Files (x86)\Microsoft Dynamics AX\60\Client\Bin"). Several files will be copied to this directory during the installation.

1.1.3 Installation and Set-up of the Lotus Notes client

In order to use the integration kit, the Lotus Notes client must have been installed on the workstation. Lotus Notes must be in a 32-bit version. For installation of the Lotus Notes client, follow the Lotus Notes installation guide.

Installation of the Lotus Notes client creates the files, which will be used by the integration kit for creation of link to Lotus Notes.
*It is **not** a requirement that the Lotus Notes client is running on the work station for AX to communicate with Lotus Notes.*

All integration with Lotus Notes is from Dynamics AX. No special modifications in Lotus Notes are required for the integration kit to be used. The integration kit may be combined with ODBC access from Lotus Notes to AX if wanted.

Also refer to the file readme.txt on the supplied media for new version specific information.

1.2 Files

The integration kit consists of the following files, which are automatically copied into the AX bin- directory of the workstation:

ERP2LN.DLL
ERP2LN_ADDON.DLL
ERP2LN.INI

BORLNDMM.DLL

1.3 Installing the Integration Kit

Insert the CD-ROM with the title “INTOGRATE Axapta – Developer”. If the installation program does not start automatically, double-click install.exe at the root of the CD-ROM (Select ’Run’ in the Windows Start menu and type ”D:\setup.exe”, if your CD-ROM drive is the D-drive).
If the installation is not carried out from the CD-ROM, the files on CD-ROM must be copied to either a local drive or a mapped network drive.

1.4 Serial number

The Integration Kit is supplied with a code letter which contains a serial number matching the license name and serial number of AX.
This serial number is supplied in a file – erp2ln.lic, that must be placed in the same directory as erp2ln.dll.

1.5 Installation of AX elements

The necessary AX files are placed in a sub-directory below the AXAPTA bin files - ”bin\Intoint\AXAPTA\x.x” (x.x is the version into which you are installing the Kit).

Start the AX client. In the menu ’Files’, select ’Open’ and then ’Project’. Then select the ’Import’ icon from the project list.

Import the file I2I_LN.xpo.

2. Quick start

Before an in-depth description of the facilities is given, a small case will be presented in this section which illustrates how the kit functions and can be used.
The case should primarily be regarded as an example of the principles in integration. The case is fully operational, but for example error handling is not incorporated. See section 4 for a more detailed description of the error codes of the various methods.

2.1 Case: Update CustTable

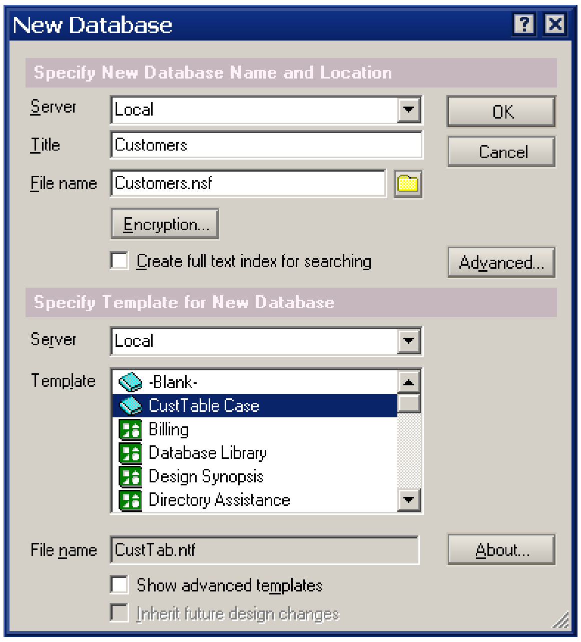
In this case it will be illustrated how data can be created and updated in Lotus Notes from AX.
The purpose of the job is to maintain a copy of the customer table in Lotus Notes. By execution of the AX job the Customer table in AX is searched, and each customer record is replicated to Lotus Notes.

2.1.1 Creating the Customer Database in Lotus Notes

The installation CD contains a template for creating a simple customer table in Lotus Notes.

The table is created as follows:

1. Copy the file CustTab.ntf from the diskette to your local Lotus Notes client data directory.
2. Start up your Lotus Notes client.
3. Create a new database [CTRL] + [N].
4. Fill out the dialog as shown below.
5. It is important that "Department" is specified as Template.
6. Select OK.


New Database

Specify New Database Name and Location

Server: Local
Title: Customers
File name: Customers.nsf

Create full text index for searching [ ]

Specify Template for New Database

Server: Local
Template: -Blank-, [✓ CustTable Case], Billing, Database Library, Design Synopsis, Directory Assistance
File name: CustTab.ntf

Show advanced templates [ ]
Inherit future design changes [ ]

2.1.2 AX code

The code for this case is contained in the file IntKitCase.xpo (located in the directory you installed INTOGRATE AX (<AX Client-bin directory>\intoint\documentation\IntKitCase.xpo).

```
static void IntKitCase(args a)
{
    ax2ln       ax2ln = new ax2ln();
    CustTable   CustTable;
    ;

    ax2ln.Init();
    ax2ln.OpenServer('LOCAL');
    ax2ln.Logon('Customers.nsf');
    ax2ln.Form('CustTable');
    if (ax2ln.GetNotesText() != '')
        return;

    while select CustTable
    {
        ax2ln.GetUnique('AccountNumber',CustTable.AccountNum);
        if (StrLen(ax2ln.GetNotesText())<=2)  // Not found
        {
            ax2ln.CreateNew();
            print CustTable.AccountNum," ",CustTable.Name,' created';
        }
        else
            print CustTable.AccountNum," ",CustTable.Name,' modified';

        ax2ln.SetFieldValue('AccountNumber',CustTable.AccountNum);
        ax2ln.SetFieldValue('Name',CustTable.Name);
        ax2ln.SetFieldValue('Address',CustTable.Address);
        ax2ln.SetFieldValue('Country',CustTable.Country);
        ax2ln.SetFieldValue('Attention',
              EmplTable::Find(CustTable.ContactPersonId).name);
        ax2ln.SetFieldValue('Phone',CustTable.Phone);
        ax2ln.SetFieldValue('Fax',CustTable.TeleFax);
        ax2ln.Commit();
    }
    pause;
}
```

2.1.3 Initialization

First the object used for the communication to Lotus Notes is prepared:

```
Ax2ln = new Ax2ln();
```

And a reference to the customer table is created:

```
CustTable   CustTable;
```

Hereafter after the integration is initialized:

```
Ax2ln.Init();
```

We specify “Local” to identify, that the database is placed on the local client:

```
Ax2ln.OpenServer(’LOCAL’);
```

Then we create the connection to the Lotus Notes database and logs on:

```
ax2ln.Logon('CustTab.nsf');
```

We specify that the design-information (e.g. field-names) should be fetched from the form "CustTable":

```
      ax2ln.Form('CustTable');
```

Now we check whether the connection has succeeded:

```
      if (ax2ln.GetNotesText() != '') return;
```

2.1.4 WHILE SELECT LOOP

The customer table is traversed in a normal way:

```
      while select CustTable
```

2.1.5 Update or Create?

Next step is to decide whether to insert a new document in Lotus Notes or to update an existing document. We do that by checking whether we can find an existing document fulfilling the unique key:

```
ax2ln.GetUnique('AccountNumber',CustTable.AccountNum);
if (StrLen(ax2ln.GetNotesText())<=2)  // Not found
{
    ax2ln.CreateNew();
    print CustTable.AccountNum," ",CustTable.Name,' created';
}
else
    print CustTable.AccountNum," ",CustTable.Name,' modified';
```

The method ***GetUnique*** retrieves a document in Lotus Notes by a search-key (AccountNum in this case). If there is only one document matching the key, a reference (DocId) to the document is returned – otherwise an error-code is returned. A reference is always identified by a string of more than 2 characters. So in this case, we check the length of the result-code and decide whether a valid document has been found. If not we decide to create a new document with the method ***CreateNew***[2].

2.1.6 Specification of Document Contents

Now we have a valid document (new or existing).
We then insert the values for some of the fields. The updated document is saved with the ***Commit*** method:

```
ax2ln.SetFieldValue('AccountNumber',CustTable.AccountNum);
ax2ln.SetFieldValue('Name',CustTable.Name);
ax2ln.SetFieldValue('Address',CustTable.Address);
ax2ln.SetFieldValue('Country',CustTable.Country);

ax2ln.SetFieldValue('Attention',EmplTable::Find(CustTable.ContactPersonId).name);
ax2ln.SetFieldValue('Phone',CustTable.Phone);
ax2ln.SetFieldValue('Fax',CustTable.TeleFax);
ax2ln.Commit();
```

[2] Under normal circumstances we would check the error-code and only create a new document if the error-code told that an existing document could not be found. If other errors where returned these should be handled in a proper way.

3. Using the Integration Kit

This section describes how the integration kit is to be used in connection with coding in AX. In section 4 the individual commands will be described in more detail.
A few more specialized methods are only described in section 4.

3.1 Call of the AX methods

To be able to access Lotus Notes from AX, an object of class ax2ln must be defined. This can be done with the statement:

```
ax2ln  ax2ln = new ax2ln();
```

The methods of the class ax2ln can be divided in two categories: Methods that returns a value directly and methods that return a value indirectly.

An example of the first type is the method ***ServerName***. This method directly returns the name of the current selected server. An example is shown below:

```
print ax2ln.ServerName();
```

An example of the second category is ***DeleteCurrent***. This method does not return any result directly. A value is returned indirectly in an internal variable. This variable contains a result-code from calling the DLL and can contain an error code. The value of the variable can be fetched by the method <u>***GetNotesText***</u>. In most cases a value of “” (blank) indicates no error calling the last command. A non-blank value normally specifies an error-code. Se section 4 for a further description of the result-codes.

In this section all methods are identified by ***bold italic***. The first reference to a method is identified by <u>***bold, italic and underscore***</u>. All methods are described further in section 4.

3.2 Connecting to Lotus Notes

Some of the integration kit commands concern the creation and closing of sessions with Lotus Notes. Other commands are for the selection of application elements in Notes.

3.2.1 Initialization of Link to DLL File

The <u>***Init***</u> method creates a link between AX and the DLL library handling the system calls to Lotus Notes. This method must be called before the other integration methods can be used. Calling the method more than once will not cause problems. After the first call of the method, subsequent calls are ignored.

On call of the ***init***-method, link is <u>not</u> created to Lotus Notes. Link is not created until the method ***Logon*** is called.

3.2.2 Creation of Session to Lotus Notes

To communicate with Lotus Notes, a Lotus Notes database must first be selected, and login can be made.
If access is made to a Lotus Notes database which is not located on the Local workstation ("Local"), a password normally must be entered (in certain cases a password will also be required for "Local" databases). This needs only be done once. The password to be specified corresponds to the current

Notes user-id password. This can be set up via the Lotus Notes client. Specification of the current Notes user is registered in the NOTES.INI file. If the user chooses to change the current ID in Lotus Notes `(/File/Tools/Switch ID)`, this ID password must be used on the next login from AX. Generally, the integration kit uses the current Notes setups.

If e.g. a current location with no link to a Notes Server has been specified ("Island*")*, it is not possible to access the Lotus Notes servers from AX either. In this case the Lotus Notes client must be started up and the location changed in order to get access to the servers.

The user rights applying to access to Lotus Notes from Dynamics AX are identical to the current rights of the ID user.

3.2.2.1 Selecting Notes server

Specification of the Notes server is by the methods ***OpenServer*** or ***OpenDialogue.***

By the first method the server name is specified as Parameter. The other method makes it possible to browse between servers and directories (as on opening of databases in Lotus Notes). As the link on table level between AX and Lotus Notes most often is unchangeable, the first method is most common.

When using the method ***OpenDialogue*** the selected server and database can be fetched by the methods ***GetDialogServerName*** and ***GetDialogDatabaseName***. Often the method ***Logon*** is called afterwards to overrule the selection of database – in other words only using ***OpenDialogue*** to select server.

Is “Local” used as servername databases on the local workplace are used.

Example:

```
ax2ln.Init();
ax2ln.OpenServer('local');
```

3.2.2.2 Selecting Notes database

After specification of server, the Notes database wanted is to be specified. As mentioned above, this can be done by interactive selection with the method ***OpenDialogue*** or by direct specification of the database name in the method ***Logon***. In the latter case the database file name is specified as parameter. *Remember to type two backslashes when specifying subdirectories.*

Example:

```
ax2ln.Logon("mail\\jdoe.nsf")
```

Apart from specifying a work database in Lotus Notes, the method also creates the actual link to Lotus Notes. It is checked whether access to server and database is possible. Furthermore the method may require a password from the user if the database is located on a Notes server, and a password has not previously been entered.

3.2.2.3 Selection of Notes Application Elements

For exchange of data between AX and Lotus Notes, either a Notes view or form must be selected. Notes view is used in connection with execution of Notes queries, whereas Notes forms are used for processing of fields.

All data processing methods require that current Notes view and/or current Notes form have been selected.

For selection of application elements, the methods ***Form*** and ***View*** are used.

```
ax2ln.Form("Memo");
ax2ln.View("Stationery"); (or ax2ln.View("($All)");)
```

3.2.3 Closing the session

If alternating sessions to Lotus Notes from AX are wanted, one session per element must be created. At the end of the program code, the current session must be closed by the command ***Close***.

This method does not interrupt the link to the DLL library which was created by the method ***Init***.
After use of ***Close***, a new link with ***Logon*** must be created on the next access to Lotus Notes.

3.2.4 Switching sessions

It is possible to operate with multiple simultaneous sessions. Each session has its own connection to Lotus Notes, with server, database, form, view, current document and more.
To change between sessions use the command ***SetHandle***.

Be cautious when using multiple sessions. Confusion about which session is active is often responsible for errors.

Use the command ***GetHandle*** to get the current session number.

3.3 Current Values

Some of the methods in the integration kit hold information on various current values. Some of these values concern the current session with Lotus Notes as was described in the preceding section.
All methods return values directly.

3.3.1 Information on Current Session

The following list shows the methods returning this type of information:

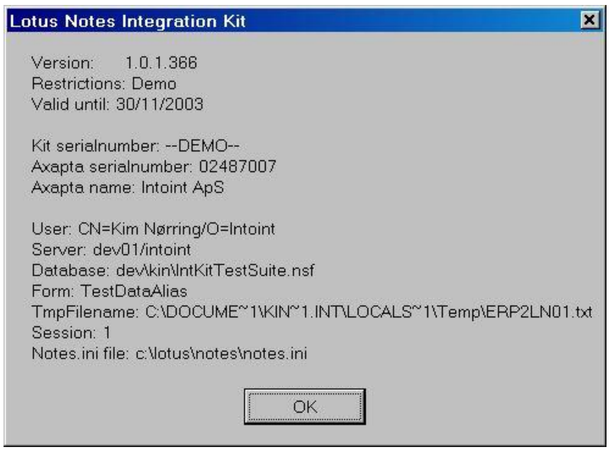
ServerName
DatabaseName
FormName
ViewName
ShowAll

The methods correspond to the methods creating the links.
Example:

```
ax2ln.OpenServer('Local');
print ax2ln.ServerName;
```

will print "Local" on the screen.

Showall displays miscellaneous information regarding the current session in a popup window.



Lotus Notes Integration Kit

Version: 1.0.1.366
Restrictions: Demo
Valid until: 30/11/2003

Kit serialnumber: --DEMO--
Axapta serialnumber: 02487007
Axapta name: Intoint ApS

User: CN=Kim Nørring/O=Intoint
Server: dev01/intoint
Database: dev\kin\IntKitTestSuite.nsf
Form: TestDataAlias
TmpFilename: C:\DOCUME~1\KIN~1.INT\LOCALS~1\Temp\ERP2LN01.txt
Session: 1
Notes.ini file: c:\lotus\notes\notes.ini

3.3.2 Information on DLL Program Library

The method ***DLLVersion*** returns the version number of the DLL library.

The command ***IsDemoVersion*** lets the user find out if the programs are currently integrated via a limited demo version or a full version of INTOGRATE.

3.3.3 Managing temporary files

On receipt of more documents from Lotus Notes temporary files in the form of comma-delimited files, are used as media.
The files have the following structure:

Line 1	"DocId", "fieldname1", "fieldname2",...,"fieldname i"
Line 2, first document	"DocId 1", "field1 value", "field2 value",..., "field i value"
Line 3, second document	"DocId 2", "field1 value", "field2 value", ..., "field i value"
...	...
Line n, last document	"DocId n", "field1 value", "field2 value", ..., "field i value"

As appears, the first part of the comma-delimited file is a list of the fields in the selected form. All fields are returned in the comma-delimited file. When the comma-delimited file is loaded into AX, superfluous fields may then be ignored. Also the methods ***ClearQueryFields*** and ***AddQueryField*** can be used to delimit the fields returned in the result set.
Apart from containing all fields, the file also specifies a unique key (DocId) for the document. It can be used for registration of documents for later selection (see section 3.4.6). The DocId field is the first value in each line.

The method ***SetTmpFileName*** is used for allocation of name to a temporary file.
If a Parameter for the method is specified, this will be used as file name.
If no Parameter is specified, a unique file name will automatically be computed or read from erp2ln.ini.
*Note that the temporary file will only be created in connection with execution of the methods **Query** or **QueryRichTextField**.*

Remember to delete the temporary file after use - use e.g. the method ***DeleteTmpFile*.**

Using the command ***GetTmpFileName***, the user can find the name of the current temporary file in the current session.

3.4 Query and lookup in Lotus Notes

There are many methods for making queries in Lotus Notes:

Query
QueryField
QueryUnique
QueryView
GetUnique
SelectID
SelectUNID
SearchView
SearchNext

These methods have different fields of application as described below.

3.4.1 General Query

The method ***Query*** is used for execution of a standard Notes query. The selection criterion is specified as Parameter. The Parameter can be compared to "View selection" in Lotus Notes. In order to use this method some knowledge of Lotus Notes is required. The command "SELECT" may be omitted when calling the method.
Here are some examples of call of the method:

```
ax2ln.Query('Select @ALL')
```
returns all documents in the database.

```
ax2ln.Query("Quantity>100")
```
returns all documents in which the field "Quantity" has a value higher than 100.

By using the method ***QueryView***, all documents in current view are returned.
Search results are always returned in a result-set in memory. Furthermore the result can be returned in a comma-separated file for further processing (see section 3.3.3). Therefore the results can subsequently be processed via the comma-reading functions in AX (CommaIO).

By using the ***Query*** method very flexible searches in Lotus Notes can be performed. But this method uses full-text queries in Lotus Notes and on larger databases this method can be quite slow. An alternative lookup method is available in the method ***SearchView*** (see section 3.4.4).

3.4.2 Handling sets of results

It is also possible to process the result of the query without using comma files. By means of various "move" commands, it is possible to specify the first, the next, the previous and the last command of the query. The methods are:

MoveFirst
MoveNext
MovePrev
MoveLast

The method ***GetFoundDocs*** informs the user of the number of documents in the result.

It is a good idea to combine this method for handling queries with a call of ***SetTmpFileName*** with an empty file name as the parameter. This way, the result will not be written to a comma file as well.

3.4.3 Query on Unique Document

With the method ***QueryUnique*** a query corresponding to the above Query, can be made which returns one document directly. To use this method it must be certain that there is only zero or one document that can be returned. However, this is often the case when using integration with AX, which almost always has at least one unique identification of a record.

3.4.4 Query by view

Since in the integration kit, queries are full text searches, this method can be quite slow in connection with large amounts of data. In this case it is better to do query in Lotus Notes using the available views (corresponding to the index in AX). However, a view can only be used for a query if all the columns searched are of the type "text" and sorted. The first search value is searched for in the first column. If needed, the second search value is searched for in the second column, etc.

The method ***SearchView*** is used for queries by search value. The method returns the first document where the document matches the search key(s),
If the user wants to specify several search keys these must be separated by a semicolon, e.g. SearchView(´Smith; John´).

```
ax2ln.View('MyView');
ax2ln.SearchView('John Doe;Sales Department');
```

If the user wants to se a different separator, it can be changed by the method ***SetDelim***.
Queries by view can also be carried out by use of the method ***SearchFirst***, which selects the first (top) document of the current view.
Using ***SearchNext*** the user can go to the following documents. This way, a view can be traversed as shown in the following example:

```
ax2ln.SearchFirst();
while (StrLen(ax2ln.GetNotesText)>2)
{
    print "Name: ", ax2ln.QueryField('Name');
    Ax2ln.SearchNext();
}
```

Notice, that ***SearchNext*** does only return documents matching the search-key(s) after ***SearchView***.
After ***SearchFirst*** all documents in the view can be traversed with ***SearchNext***.

3.4.5 Query on Field Value

Another type of query is that on field values of a current document. This is performed by the method ***QueryField***.

To be able to use this method, a current document must have been selected, e.g. by one of the methods ***SelectID***, ***SearchFirst***, ***GetUnique*** or others.

Example:

```
ax2ln.GetUnique("Number", Number);
print Number,": ",ax2ln.QueryField("Name");
```

QueryField always returns a text value (STR). Thus it is the task of the AX programmer to convert the value to the correct AX format, e.g. by Str2Num or Str2Date. This area of course requires a certain level of knowledge of the current Lotus Notes formats, e.g. date values etc.

In connection with queries in rich text fields, the text may be very long. ***QueryRichTextField*** can be used for handling this situation. It works like ***QueryField*** and will return a value in the same way – directly in the call. But in addition, a the field value will also be written to the current temporary file, e.g. to be loaded into AX.

Example:

```
AsciiIO fileRichText;
str strRichLine;
...
ax2ln.QueryRichTextField('RichTextValue');

fileRichText = new AsciiIO(ax2ln.GetTmpFilename(), 'R'); // Open temporary file.

if(fileRichText) {
  fileRichText.inRecordDelimiter('\r\n');

  [strRichLine] = fileRichText.read(); // Read first line from file
  while(fileRichText.status() == IO_Status::OK) {
    print 'Data: ', strRichLine; // Print line
    [strRichLine] = fileRichText.read(); // Read next line from file
  }
}
```

Use of ***QueryField*** requires the field to be defined in the current form. Under certain circumstances, there are items in Notes documents, which are not defined in the form. In such cases, the method ***GetFieldValue*** can be used since it does not check the field definition.

3.4.6 Unique Query

For data to be processed in a relational database, unique selection of all data must be possible. This also applies to AX. But Lotus Notes is not a relational database, and Notes has not been designed to support unique keys as has AX for example.
But for integrating AX with Notes, it is necessary to be able to make unique queries.
The integration kit has three methods for handling this: ***SelectID, SelectUNID*** and ***GetUnique***. All return a current document which may then be processed by e.g. ***QueryField*** or ***SetFieldValue***.

GetUnique is used in situations when there is a Unique index-key in AX which can make a unique selection of document in Notes. This might e.g. be "Account number" in the chart of accounts that will return one account. But for example it cannot be "Account number" in ledger transactions as many transactions have the same account number.

Example:

```
ax2ln.GetUnique('Number', Number);
print Number,": ",ax2ln.QueryField('Name');
```

GetUnique has two Parameters. The first Parameter is a field name; the second Parameter is a field value. The method will either return an error value or a DocId. Normally error values are found in the interval "1" to "99". If the return value (***GetNotesText***) has more than two characters, it is a reference to a document. The document is active immediately. Subsequent query via ***SelectID*** is not necessary.

SelectID is used in situations when direct query via a known DocId is wanted. This DocId might be produced as the result of a Query which returns DocId in the comma-delimited file.
DocId is unique to the database. But the value can be "re-used" after deletion in the Notes database. Furthermore DocId is only unique to the individual instance of the database. *In connection with replication or "Cut-and-paste", DocId may be changed. **SelectID** will therefore not necessarily give the same results on other replica of the database. Instead use **GetUnique** (refer to Appendix 2.6).*

To avoid this problem, the Lotus Notes Universal Document ID (UNID) with the method ***SelectUNID*** can be used for the query. The use is analogous to SelectID, but UNID is unique across replicas of the same database. In return, UNID is made up of 32 characters whereas a DocID is made up of 8 characters.

3.4.7 Comparison of Values

Often there is a need for being able to compare field values in Lotus Notes and Dynamics AX. It is especially necessary on update of existing documents in Notes by which unnecessary update of a field may result in an "unread" mark being set. This can be avoided if updates are only carried out in case of differences between Notes and AX. The functionality can be obtained by reading the Notes value by ***QueryField*** and subsequently comparing the values. By the method ***CompareField*** the entire comparison can be performed by one command.
An example of the use of this method is shown below:

```
...
if (ax2ln.CompareField('Account',LedgerTable.AccountNum) != '')
{
    ax2ln.SetFieldValue('Account',LedgerTable.AccountNum);
    ax2ln.Commit;
}
...
```

An alternative is to specify that in connection with all data transferred to Lotus Notes, the “old” value must be compared with the “new” value. And only where these values differ will the new value be saved in Notes. This way, unnecessary updates with resulting setting of e.g. unread marks can be avoided. However, this method can be a bit slower.
The command ***SetKeepUnread***(TRUE) activates this feature.

3.5 Data Manipulation

After establishment of a session with Lotus Notes and maybe selection of a document for processing, data can be processed by means of different methods. These methods make it possible to create, edit or delete documents.

3.5.1 Creation of a New Document

The method ***CreateNew*** creates a new empty document in Lotus Notes.
After call of ***CreateNew*** the field values should be filled in by the method ***SetFieldValue***.

Remark, that the document is both created and committed. An empty document will therefore exist after calling ***CreateNew***.

3.5.2 Allocation of Field Values

SetFieldValue is one of the most essential methods in the integration kit. By this method the values of the fields in Lotus Notes documents are changed.
The method is called with at least two Parameters: field name and field value. If a third Parameter is set to TRUE, this specifies an immediately update to Lotus Notes. Alternatively ***Commit*** is called after allocation to all fields.
Field value must always be a text string or text variable. Thus the AX programmer should convert other field types to a string before the call.

Example:

```
ax2ln.SetFieldValue(‘Date’,Date2Str(LedgerTrans.TransDate,123,2,3,2,3,2));
```

In certain situations the order in which the fields are filled in may be of importance. Therefore it is recommended to fill in the fields in the same order as if the fields were to be filled in via the Notes form.

SetFieldValue can also be used for rich text fields, but the contents of the field will be replaced each time the method is called. However, more lines can be added by entering a new line (Num2Char(10)) in the text to be transferred as a parameter.

If the contents of a file are to be inserted in a rich-text field, the method ***ImportFile*** is used. The application of this method corresponds to that of ***MailBodyFile***. See section 3.8.2 for a detailed description of import of files.

If the need to set a field value in an item where no field is defined in the form arises, the method ***PutFieldValue*** can be used. Since this method is intended for situations when the field is not defined in the form, placing the field name first in accordance with the following pattern:

#Datatype#, e.g. “`ax2ln.PutFieldValue(‘#DATETIME#Date’,D)`”.

If no field type is specified and the field is not present on the form-design the data will be saved as a text type.

Data transferred to DateTime fields/items must always be formatted as: “dd-mm-yy” or “dd-mm-yyyy”. That is delimiting day, month and year by a dash and with day first and year last.

Is there a need to append test to a field/item, where there is already data present, the method ***AppendFieldValue*** can be used. The method corresponds to the method ***SetFieldValue***, but with ***AppendFieldValue*** the existing content in the field/item is kept. If this is used for multi-value fields remember to delimit values with semi-colon.

```
ax2ln.SetFieldValue('KeyWord',"Value 1");
ax2ln.AppendFieldValue('KeyWord',";Value 2");
ax2ln.AppendFieldValue('KeyWord',";Value 3");
```

The method can also be used for transferring multiple lines to a rich-text field in Lotus Notes. Here lines are delimited by Num2Char(13)+Num2Char(10).

A variant of this method is ***AppendTextList*** that is used for entering data to multi-value fields (and only to multi-value fields). Here semi-colon shall not be used as delimiter.

3.5.3 Deletion of Documents in Lotus Notes

The integration kit features two methods that enable deletion of documents in Lotus Notes.

DeleteID can be used for deletion of a document by specifying DocId. By this method it is not necessary first to retrieve a document as current document.

Similarly, ***DeleteCurrent*** is used for deletion of a current document which has e.g. been found via the method ***GetUnique***.

After deletion of a document via **DeleteCurrent***, a new current document must be selected before further data manipulation can be made with methods that process the current document (e.g.* **SetFieldValue***).*

3.5.4 Updates of changes

In AX the concept TTS is applied. By TTSBEGIN, TTSABORT and TTSCOMMIT data updates can be controlled and e.g. aborted. The integration kit features a facility that corresponds to this system. However - only one document can be handled at a time.

To update changes, the method ***Commit*** must be called afterwards.

This must be done before another current document is selected and before the current session is closed.
Similarly, this may be utilized to cancel changes by not applying ***Commit***.

A variant of ***Commit*** is ***CommitNoRecalc*** which also saves the current document, but which does not create an item for each field on the current selected form.

Commit and ***CommitNoRecalc*** saves the document without calculating formulas on fields and the form. If formulas should be execute the method ***CommitWithForm*** must be used. This method works in the same way, as if [Ctrl]-[S] is pressed in the Lotus Notes client to save an open document. This method can of course be slow compared to the two other methods, because of the extra data-processing. If no formulas is present on the form, ***Commit*** and ***CommitNoRecalc*** can be used to improve performance.

The method ***SetComputeWithForm***, can be called to adjust the integration kit to always re-calculate formulas when calling ***Commit*** .

The table below illustrates the difference between the three commit methods:

	Create item for each field on form	Recalculate formulas on fields and forms
CommitNoRecalc	-	-
Commit	X	-[3]
CommitWithForm	X	X

3.5.5 Information on Active Document

Using the update methods will result in error messages if a Notes document has not been selected. For checking whether a document is active, the method ***DocId*** can be used. This method returns the DocumentID of the active document.
The method can also be used after ***QueryUnique*** and ***GetUnique*** (if ***GetNotesText*** is not read immediately after call of these methods).

If desired, UNID can be used instead by calling the method ***UNID*.**

For more advanced use, further information about the current document and the individual fields can be obtained.

The command ***GetDocInfo*** provides the user with various information about the current document, including e.g. the date of creation and last editing, information about whether "parent document", etc.

Similarly, ***GetFieldLength*** and ***GetItemInfo*** can provide information about the values of the individual fields and items.

[3] If ***SetComputeWithForm*** is set to TRUE formulas will be calculated though.

3.6 Rich-text

AX does not support rich-text fields as in Lotus Notes. But there may be some situations in which it is needed from AX either to view information in Lotus Notes that is stored in rich-text fields or to store rich-text information in Notes.

3.6.1 Update to Notes

Via the integration kit it is possible to immediately store ordinary text from AX in a rich-text field in Lotus Notes. Thus it is not necessary to distinguish between the field types being updated in Notes, i.e. whether it is a "Text" or "Rich-text" type of field. The method ***SetFieldValue*** can be used for both field types.
Alternatively use the methods ***ImportFile***.

3.6.2 Reading from Notes

The same applies to reading data in Lotus Notes. Here the method ***QueryField*** is used in the normal way. The integration kit will try to return data in a format that AX can read. Rich-text fields in Lotus Notes may contain some types of information that are not "visible" in AX, e.g. pictures, attachments, formatting codes, more lines etc.
All formatting codes are removed on reading from Lotus Notes. Pictures, attachments etc. are not transferred. If a rich-text field contains more lines of text, these are returned by the integration kit as one string in which new line is marked by the ASCII value 10 (LF). Then it is the task of the programmer to separate it into more AX lines, e.g. in a note field. Alternatively the method ***QueryRichFieldValue*** can be used; whereafter the value can be read from a file.

If the method ***Query*** is used, by which the result is returned in a comma-delimited file, the method applied is slightly different. If a Notes document contains one or more rich-text fields with more lines, the information will be distributed on several lines by the comma-delimited file when the field is transferred to this file.

Example:

Number (Text)	1234
Name (Text)	John Doe
Description (Rich Text)	A customer Created with Three Lines

This document would give the following lines in the comma-delimited file:
```
....
"1234","John Doe"," A customer
Created with
Three Lines "
....
```

When this comma-delimited file is loaded to AX, it will read these three lines as **three** documents/records.

This problem can be solved in two ways. Either rich-text fields are isolated so they are not included in the comma-delimited file directly, or storage is only made of the first line of the rich-text field.

3.6.2.1 Rich-text in Separate Files

If the entire contents of the rich-text fields are to be kept, a method is chosen which saves all rich-text information in separate comma-delimited files. One file per document and per rich-text field. To have access to the contents of these fields, they must be loaded via separate loading into AX.

The separate rich-text files are stored in a subdirectory under the name "RichText".
The files are named according to the following syntax:

```
<DocId>.<FieldNumber>
```

Example: "00002E40.3" where "00002E40" is the DocId of the Notes document, and "3" specifies that it is the third field in the form.

3.6.2.2 Selection of Method

If it is chosen not to save the contents of rich-text fields in separate files, they will be transferred like other field types to the comma-delimited file. However, the contents may be abbreviated to the first line if the field has more lines.

Selection of method is by the method ***SetRichSepMode***. The current value can be read by calling the method ***GetRichSepMode***. The value TRUE specifies separate files, whereas FALSE specifies storing of rich-text values in the comma-delimited file.

Default value is storing the first line of rich-text fields in the comma-delimited file (FALSE).

3.7 File Attachments

The integration kit has various methods for the handling of attachments in Notes documents. These methods may be used for attaching e.g. a text file from an AX report to a Notes document.
However, all types of files can be attached.

Attachments are attached to a Notes document via the method ***AttachFile***. The first Parameter specifies the name of the file. Another Parameter may specify the name that the attachment is given in the document.

Example:

```
ax2ln.AttachFile('c:\\autoexec.bat','AUTOEXEC').
```

If no other attachment name is specified, the attachment will be created with the same name as the file (excluding preceding path. In the example the name would be "autoexec.bat").

Existing attachments can be detached, i.e. copied to a file, via the method ***DetachFile***. In this case too, the file name is specified as the first Parameter. This file name must not necessarily be the same as the file name of the file that was originally attached. The file can be detached with another file name. As described above, any attachment name is specified as the second Parameter.
Example:

```
ax2ln.DetachFile('c:\\autoexec.new','AUTOEXEC')
```

Here, the file of the preceding example is saved as a new file so the original file is not overwritten.

Whether the current document has attachments can be detected by calling the method ***ExistsAttach*.**

```
if(ax2ln.ExistsAttach('config.sys') {
  // Attachment exists and can be fetched
  ax2ln.DetachFile('config.sys');
} else {
  // Attachment does not exist.
}
```

Furthermore, the integration kit has methods for deletion of attachments, i.e. ***DeleteAttachment*** and ***DeleteAttachmentAll***. These methods may either be used for deleting one or all attachments.

All actions on attachments operate with attachments directly in the document. Attachments in rich-text fields are handled, but the icon representing the attachment remains on the document.

3.8 Mail

Apart from offering the possibility of exchanging data with Lotus Notes, the integration kit has a facility for **sending** mails directly from Dynamics AX.

Building code for sending mails from AX differs from the preceding methods as set-up of mail server, database and forms needs not be carried out by the user. The integration kit will by itself find the necessary information in NOTES.INI. Thus server etc. must not be specified with the methods ***Server***, ***Logon***, ***Form*** and ***View***.

3.8.1 Preparing Mail

The mail routine of the integration kit can be divided into three steps:

1. Preparation of mail
2. Building up the actual mail contents (body text)
3. Sending of mail

First, the mail is prepared. This implies specification of the following information:
- "SendTo", the E-mail address of the receiver (MUST BE SPECIFIED)
- "Subject", mail heading
- Optional "CC", any other receivers of copy (Carbon Copy)
- Optional "BCC", any other receivers of copy (hidden from the receiver - Blind Carbon Copy)

Preparation of mail is by the method ***PrepareMail***. The method receives 2-4 Parameters in the above order. CC and BCC (the third and fourth Parameter) are optional.

Call of the method might look as follows:

```
ax2ln.MailPrepare("John Doe/Acme","New","James Kirk/Acme,Ted Masters/Acme")
```

In this example two persons have been specified as receivers of copy of the mail.

The individual elements of the mail header can also be set by the methods ***MailSetSendTo***, ***MailSetCC***, ***MailSetBCC*** and ***MailSetSubject***.

3.8.2 Filling in Mail Contents

The next step is to fill in the body text of the mail (the body field).

This can be done in two ways. A simple method by which only ordinary text is filled in, and a more advanced method by which it is possible to combine text and contents of e.g. ASCII files, rich-text files (RTF) and/or pictures (certain formats).

The simple method is used for creating simple mails that consist of ordinary body text, without the need for formatting etc. This method is faster than the advanced one.
The body text is built up by calling the method ***MailBodyText***. The method may be called an arbitrary number of times. Each call adds a new line to the mail. Mails built up by this method should be readable by all mail systems. This is especially interesting in cases where the receiver does not apply Notes mail but e.g. Internet mail.
Many of the facilities of the advanced method will often not be readable by other mail systems (and if they can be read, the formatting codes used are most often left out anyway).

The more advanced method is used via two methods:
MailBody which is used analogously to the above method (***MailBodyText***) and ***MailBodyFile*** which is used for merging the contents of a file.
The facility for merging e.g. rich-text files (RTF) makes it possible to format the contents of the mail, e.g. with various fonts. However, this requires that the contents be built up by means of rich-text codes. Contact Intoint for more information on this.

Both methods can be supplemented with a facility for attaching files to mails, i.e. via the method ***MailAttachFile***. This method corresponds to ***AttachFile*** (see section 3.7).

3.8.3 Sending of mail

When the mail has been prepared, it can be sent. This is simply done by the method ***MailSend***.

A Parameter in the form of a file name may be specified. If such a Parameter is specified, the contents of the specified file are entered in the body field.
Existing contents that have been filled in via the methods mentioned in the preceding section are kept.

Below are shown two examples of AX jobs sending a mail:

Example 1:

```
ax2ln    ax2ln = new ax2ln();
;
ax2ln.init();
ax2ln.MailPrepare('abc@acme.dk','Congratulation');
ax2ln.MailSetBodyText("Happy birthday");
ax2ln.MailSend();
```

Example 2:

```
ax2ln    ax2ln = new ax2ln();
;
ax2ln.init();
ax2ln.MailPrepare('Support@acme.dk','Error');
Ax2ln.MailSetBodyFile("c:\\pictures\\logo.bmp");
ax2ln.MailSetBody("");
ax2ln.MailSetBody("I got this error:");
ax2ln.MailSetBodyFile("c:\\temp\\scrndump.bmp");
ax2ln.MailSetBody("");
ax2ln.MailSetBody("Please help me out");
ax2ln.MailSetBodyFile("c:\\pictures\\sign.bmp");
ax2ln.MailAttachFile("c:\\AX\\MyProject.xpo");
ax2ln.MailSend();
```

3.8.4 Mail through Lotus Notes client

The approach described above handles sending mails by coding - without user-interaction.

With the command ***MailComposeOLE*** it is possible to activate the Lotus Notes mail-client ready to send a new mail. The mail-fields SendTo, CC, BCC and Subject can be prepared as described above.

3.8.5 MAPI Mail

If the company does not use Lotus Notes for mailing (but e.g. only as a Lotus Domino Web server) it is possible to use the MAPI protocol for simple mails (without attachments, etc.). Microsoft Outlook uses the MAPI protocol among others.

MAPI mailing can be activated by using ***MailSetMAPI***(TRUE).

If needed, the methods ***MAPIProfile*** and ***MAPIPassword*** can be used to specify the current users MAPI profile and passwords. This will prevent boxes prompting the user for such information.

Please, notice that not all Lotus Notes mail features are available in this integration kit for MAPI.

3.8.6 Automatic Error Messages via Mail

The integration kit has a facility by which messages on errors occurring in connection with integration are mailed to e.g. the person responsible for IT. This may facilitate internal support as the mail contains most of the relevant information. This facility is especially well suited when using the integration kit on a batch server where error messages may not necessarily be seen immediately.

The facility is activated by the method ***MailErrorTo*** in which the receiver is specified as Parameter. More receivers can be specified, separated by commas. If e.g. the following AX code lines are specified in the beginning of an AX script, John Doe will automatically be informed of all error situations.

```
...
ax2ln.init();
ax2ln.MailSetError("John Doe/Acme")
```

An example of a mail with error message is shown below:

```
To:           John Doe/Acme
cc:
Subject:   ERP2LN ERROR: Form (2): Form werjkhewkjr does not exist
Error: Form (2): Form werjkhewkjr does not exist
Date: 19/8/03 13:16:40
DLL Version: 1.0.1.143 - 18.08.2003
Handle: 2

User: CN=Flemming Nielsen/O=Intoint
Server: dev01
Database: Action\Department
Form:
View:
DocId: 0
```

3.9 Information about Notes design

In certain situations the user may want to query whether a field, form or view exists in a Notes database, or the user may want a list of the existing design elements.

The integration kit holds a number of methods capable of supplying this information. Queries can be made on names of fields, forms and views.

The method is the same for all three types of design elements. To be able to query, the query basis must be prepared. For example, to make form queries, the user must first log on to the database. To make field queries, the form must first be selected.

3.9.1 Query on the Existence of Design Element Name

It is possible to query on the existence of design elements by the following methods:
ExistsField, ***ExistsForm*** and ***ExistsView***.

If the value TRUE is returned, the design element exists in Notes. If FALSE is returned, it does not exist. Error values can be retrieved by ***GetNotesText***.

3.9.2 Listing Notes Design Elements

The integration kit has 6 methods for listing e.g. fields in a form or views in a database, i.e.

FirstFieldName and ***NextFieldName***
FirstFormName and ***NextFormName***
FirstViewName and ***NextViewName***

A printout of all fields in a form could e.g. be designed as follows:

```
str 40  f;
ax2ln   ax2ln = new ax2ln();
;

ax2ln.init();
ax2ln.OpenServer('LOCAL');
ax2ln.Logon('mail\\xxx.nsf');
ax2ln.Form('Memo');

f = ax2ln.FirstField;
print "Fields on Memo form in Lotus Notes”;
while (F != '')
{
    print f;
    f = ax2ln.NextField;
}
pause;
```

A similar script can be made for forms and views.

3.9.3 Type of Design Element

The integration kit also has a facility for giving information about the types of the various design elements.

The method ***FieldType*** returns the type of a Notes field, e.g. "RichText" or "Number".

The method ***FormType*** returns the type of the current Notes form: "Main", "Response to Main" or "Response to Response". To use this method, the form must have been selected by the method ***Form***.

Below is an example of a script listing forms and their types in the Notes help database:

```
STR 40  f;
ax2ln   ax2ln = new ax2ln();
;

ax2ln.init();
ax2ln.OpenServer('LOCAL');
ax2ln.Logon('help4.nsf');

f = ax2ln.FirstForm;
print "Forms in the database help4.nsf”;
WHILE (f != '')
{
    ax2ln.Form(f);
    print f,": Type = ",ax2ln.FormType;
    f = ax2ln.NextForm;
}
pause;
```

At present the integration kit has no analogous functions for Notes Views.

However, the kit does hold the method ***DatabaseTitle*** which returns the full name of the current data-base (not only the file name (see ***DatabaseName***)).

3.10 Debugging

The integration kit has various facilities for solving problems when developing applications with integration between Dynamics AX and Lotus Notes.

Most methods return a return-code that can be examined through ***GetNotesText***. This variable can then be tested after the call and any errors can be dealt with.

Here is an example:

```
ax2ln.Init;
ax2ln.OpenServer(Server);
if (ax2ln.GetNotesText !='')
    return 0;
```

3.10.1 Error messages

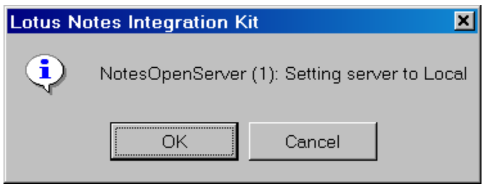
When an integration task has been defined and tested, errors will occur only rarely (e.g. if the Domino server is shut down). Therefore, error messages from the DLL will usually appear in a message box.
If a run is not to be monitored and thus NOT to be interrupted by error messages, the error messages can be de- or re-activated by the command ***Messages***. If error messages are de-activated extra care should of course be taken when dealing with areas of the AX code where errors may occur.
The command ***GetMessages*** can be used to find out if error messages have been activated.

3.10.2 Debugging on the screen

Since Dynamics AX communicates with Lotus Notes through a DLL it is not possible to see what happens from the point when AX makes a call to Lotus Notes and until the call returns.

The method ***Debug*** activates a function which shows details from the DLL which is integrated with Lotus Notes. This information comes directly from the DLL and is therefore shown in separate Windows windows. The function is activated by ***Debug***(TRUE) and de-activated by ***Debug***(FALSE). The function can also be de-activated by selecting "Cancel" in the debug windows.
Activation of Debug affects all sessions – not just the current one.

Below is an example of information from the DLL:


Lotus Notes Integration Kit

NotesOpenServer (1): Setting server to Local

3.10.3 Debugging to log file

If a system is to be monitored and it is not possible to monitor the screen regularly or if the debugging contains too much information to handle through the dialogue boxes, the debug information can be sent to a log file.

The name of this log file can be specified by using the command ***LogFileName***. ***ClearLog*** clears the log file.
SetLLog(TRUE) activates debug logging in a file while ***SetLog***(FALSE) deactivates it.

Activation of logging affects all sessions.

4. Reference

This section describes the integration kit methods in the class ax2ln in detail.
Some sections may be omitted for some methods.
The description of each method follows the structure below:

Method name

Syntax

Description of how the method is called.

Parameters are written in brackets (< >). The contents between the brackets must be substituted by the current value.
For example, `DeleteID(<DocId)` may in practice look as
`DeleteID(‘00003A08’)` or `DeleteID(ID).`

Optional Parameter values (optional use of Parameter) are specified in square brackets ([ ]). For example `SetTmpFileName([<File Name>])` can be called with both the method `SetTmpFileName("Temp.$$$")` and just `SetTmpFileName().`

Input

Input Parameters. Type is specified in parenthesis

Output

Any output Parameters.
If a value is returned directly (function) this is specified as "direct"; otherwise there is a specification of the variable to which the return value is returned, normally via ***GetNotesText***. Type is specified in parenthesis (normally "*text*").

Function

Description of the action(s) executed by the method.

Example

Example of the use of the method.
This paragraph may be omitted if example is deemed superfluous. The examples will often only show parts of a program. Especially initialization and closing parts may be omitted.
In the examples it is typically assumed, that an object of class ax2ln has been defined:

```
ax2ln   ax2ln = new ax2ln();
```

Error codes

Any error codes. These can, and should always, be checked after call of method so that the AX code will handle any errors.
Error codes can always be read by calling the method ***GetNotesText***.

Section

Reference to section(s) in this manual where the use of the method is described.

Other references

Reference(s) to related or similar methods.

The methods are listed in alphabetical order.

AddQueryField

Syntax

```
AddQueryField(<Fieldname>)
```

Input

(Text)
1. parameter: Fieldname in Notes to be included at next Query

Output

None

Function

Specifies, that this Notes field should be included among the fields whose values is written to the comma separated file at call to ***Query***.
Fields are included in the file in the order they are entered by ***AddQueryField***.

Example

```
ax2ln.ClearQueryFields():
ax2ln.AddQueryField('Number');
ax2ln.AddQueryField('Name');
ax2ln.Query("Nummer>'100'");
```

Error codes

"1":	Field is not defined on current form
"2":	Form not selected, use method ***Form***
"3":	No active document or no fields on form
”4”:	Maximum number of search fields exceeded
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

Other references

ClearFields
Query

AppendFieldValue

Syntax

```
AppendFieldValue(<Fieldname>,<Fieldvalue>[,<COMMIT>])
```

Input

(Text, Text[, Boolean])

1. Parameter:	Fieldname in Notes where value is to be appended
2. Parameter:	Value
3. Parameter:	Optional selection of commit

Output

None

Function

AppendFieldValue adds the value supplied in the second parameter to the current values in the field supplied in the first parameter.
The value must be of type Text (convert if necessary). Existing content in the field is not overwritten by this method.

If TRUE is specified as the third parameter, the value is stored in Notes immediately. This is useful in the case of a single update.

AppendFieldValue is often used by multi-value fields. In these cases remember to delimit the values with a delimiter character (default is ";" - see explanation at ***SetListDelim***).

Example

```
ax2ln.SelectID(F.DocID);
ax2ln.SetFieldValue('Number',"Value 1");
ax2ln.AppendFieldValue('Number',"Value 2",TRUE);
```

Error codes

"1":	Field not found in form
"2":	Form not selected, use method ***Form***
"3":	Document not selected
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

Other references

SetFieldValue
AppendTextList
SetListDelim

AppendTextList

Syntax

```
AppendFieldValue(<Fieldname>,<Fieldvalue>[,<COMMIT>])
```

Input

(Text, Text[, Boolean])

1. Parameter:	Fieldname in Notes where value is to be appended
2. Parameter:	Value
3. Parameter:	Optional selection of commit

Output

None

Function

This method appends the value supplied in the second parameter to the current values in the mulit-value field supplied in the first parameter.
The value must be of type Text (convert if necessary). Existing content in the field is not overwritten by this method.

The difference between ***AppendFieldValue*** and ***AppendTextList*** is, that ***AppendTextList*** always adds a new values in a multi-value field.
Existing value are never overwritten by ***AppendTextList.***

If TRUE is specified as the third parameter, the value is stored in Notes immediately. This is useful in the case of a single update.

Example

```
ax2ln.SelectID(F.DocID);
ax2ln.SetFieldValue('Number',"Value 1");
ax2ln.AppendTextList('Number',"New multi value",TRUE);
```

Error codes

"1":	Field not found in form
"2":	Form not selected, use method ***Form***
"4":	Document not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

SetFieldValue
AppendTextList
SetListDelim

AttachFile

Syntax

```
AttachFile(<Filename>[,<Attachment Name>])
```

Input

(Text, Text)
1. Parameter: Name of file to be attached
2. Parameter: Optional name on attachment. If this parameter is not stated, the attachment will be named with the filename.

Output

None

Function

Attaches the file specified in the first Parameter on to the current document (*Note: Attachment is not made to a specific rich-text field, only to the actual document*).
The attachment will appear at the bottom of the Notes document with its attachment name.
If another Parameter is specified, the attachment will get this name in the Notes document.
This facility may e.g. be used for a more descriptive name.

Example

```
ax2ln.GetUnique('Number',"123");
ax2ln.AttachFile('c:\\autoexec.bat',"Startup");
```

Error codes

"2":	Form not selected, use method ***Form***
"3":	Document not selected
"98":	The file supplied as first parameter does not exist or cannot be read.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.7

Other references

DetachFile
DeleteAttachment
MailAttachFile

ClearLog

Syntax

```
ClearLog()
```

Input

None

Output

None

Function

Empties current log file.

Example

```
ax2ln.SetLogFileName('debug.log');
ax2ln.ClearLog();
ax2ln.SetLog(TRUE);
ax2ln.Debug(TRUE);
ax2ln.SearchView('1234');
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.10.3

Other references

SetLog
SetLogFileName
Debug

ClearQueryFields

Syntax

```
ClearQueryFields()
```

Input

None

Output

None

Function

Clear list with fields to be included in result set in next ***Query***.

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Example

```
ax2ln.ClearQueryFields();
ax2ln.AddQueryField('Number');
ax2ln.AddQueryField('Name');
ax2ln.Query("Nummer>'100'");
```

Other references

AddQueryField
Query

Close

Syntax

`Close()`

Input

None

Output

None

Function

Closes the session with Lotus Notes. After execution of the method, ***Logon*** must be exe-cuted again to be able to communicate with Notes.

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.3

Other references

Logon

Commit

Syntax

```
Commit()
```

Input

None

Output

(Text)
Returns the 8 digit hexadecimal DocId on the updated Notes Document.

Function

Update of changes in Lotus Notes. Data is not saved in Lotus Notes until this method has been executed. Only one active document can be handled. Change to a new current document will mean that changes are not updated, unless ***Commit*** is called.
Beware that formulas on fields and form is **not** executed. If this is necessary call ***CommitWithForm.***

Example

```
ax2ln.GetUnique('Number',"123");
ax2ln.SetFieldValue('Name',"Sheet");
ax2ln.SetFieldValue('Responsible',"HKR");
ax2ln.Commit();
ax2ln.GetUnique('Number',"124");
```

Error codes

”1”:	Form not selected, use method ***Form***
"2":	No data to save
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.4

Other references

CreateNew
SetFieldValue
DeleteCurrent
DeleteID

CommitNoRecalc

Syntax

```
CommitNoRecalc()
```

Input

None

Output

(Text)
Returns the 8 digit hexadecimal DocId on the updated Notes Document.

Function

Corresponds to ***Commit*** (see this). However, while ***Commit*** creates an item for each field on the current selected form, ***CommitNoRecalc*** does only create an item for each item set by the integration kit.

Error codes

”1”:	Form not selected, use method ***Form***
"2":	No data to save
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.4

Other references

Commit

CommitWithForm

Syntax

```
CommitWithForm()
```

Input

None

Output

If ***GetNotesText*** returns a 8-digits hexadecimal code, this code identifies the DocId, that the saved document has in Lotus Notes. Otherwise the return-code is an error-code.

Function

This method works as ***Commit***. But with ***CommitWithForm*** all formulas on fields and the form will be executed. This method is used, if there are fields of type “computed” on the form.
CommitWithForm can be considerably slower than ***Commit***.

Error codes

”1”:	Form not selected, use method ***Form***
"2":	No data to save
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.4

Other references

Commit
SetComputeWithForm

CompareField

Syntax

```
CompareField(<Fieldname>,<Fieldvalue>)
```

Input

(Text, Text)

1. Parameter:	Fieldname in Notes
2. Parameter:	Value in AX to compare to.

Output

(Boolean: Direct)

FALSE:	Notes Field-value is <u>different</u> than AX Value
TRUE:	Notes Field-value is <u>identical</u> to AX Value

Function

Compares the value of the Notes field selected by the first Parameter to the value of the second Parameter. Returns TRUE if the two values are identical.
Useful when reducing the number of updates, and can be used to avoid unnecessary up-dates by which "read" marks are removed.
Note that the value is read directly and therefore there is no need to call ***GetNotesText***.

Example

```
ax2ln.GetUnique('DepNumber',Number);
if (StrLen(ax2ln.GetNotesText())>2)
    if (ax2ln.CompareField("DepName",Name))
        print "Dep. ",Number," updated in Notes";
```

Error codes

"2":	Form not selected, use method ***Form***
"3":	Field is not defined on Notes form
”99”:	Error in parameters.

Section

3.3

Other references

QueryField

CopyToDatabase

Syntax

```
CopyToDatabase(<Server>,<Database>)
```

Input

(Text)

1. Parameter:	Name of desitnation server
2. Parameter:	Name of desitnation database

Output

None

Function

Copies current active document to another database. This database can be present on the current server or another server. Destination server and database must be supplied.
Use the methods ServerName and Databasename to get the names of the current server and database.

Example

```
ax2ln.SearchFirst();
while StrLen(ax2ln.GetNotesText)>2
{
  ax2ln.CopyToDatabase('Local',DataBaseName());
  ax2ln.SearchNext();
}
```

Error codes

"0":	No active document
"1":	Failed To Open Database
"2":	Failed to copy active document to database
“3”:	Error writing file to disk
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

CreateLinkFile

Syntax

```
CreateLinkFile(<filename.url>))
```

Input

(Text)

1. Parameter:	Filename on file to be created with Lotus Notes link-information (must have the extension ’.url’).

Output

None

Function

Creates a file with filename as supplied in parameter. The file contains link-information on the current selected document in Lotus Notes.
By double-clicking this file in Windows, the Lotus Notes client is launched and the document is opened.

Example

```
ax2ln.GetUnique('Nummer',’123’);
ax2ln.CreateLinkFile(‘c:\shortcut.url’);
```

Error codes

"1":	Document not selected
"2":	No filename supplied
“3”:	Error writing file to disk
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

CreateNew

Syntax

```
CreateNew([“NoCommit”])
```

Input

([Text])
optional parameter: “NoCommit” Specifies, that an empty document should <u>not</u> be created.

Output

None

Function

Creates a new empty document.
Fill in values by ***SetFieldValue*** and save the document by ***Commit***. The document is created with the current form (***Form***). If the form type in Notes is a "response-to-response document", the new form will be created as a Response document to the current document. Similarly, a "Response document" will be created as a Response document to the main document.
Note that the new document is automatically committed by this command. Thus an empty Notes document is saved initially, unless the optional parameter “NOCOMMIT” is supplied.

Example

```
ax2ln.CreateNew();
ax2ln.SetFieldValue('Number',"125");
ax2ln.Commit();
```

Error codes

"2":	Form not selected, use method ***Form***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.1

Other references

Form
FormName
SetFieldValue
Commit

DatabaseName

Syntax

```
DatabaseName()
```

Input

None

Output

(Text: Direct)
Filename for current Notes database

Function

Returns the current database filename that has been set by ***Logon*** or ***OpenDialog***.

Example

```
PRINT "Current database: ",ax2ln.DatabaseName();
PAUSE;
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.3.1

Other references

Logon
OpenDialog

DatabaseReplicaId

Syntax

```
DatabaseReplicaId()
```

Input

None

Output

(Text: Direct)
Replica ID of current database.

Function

Returns the 16 character replica id of the current database

Example

```
info(StrFmt("Database: %1 ",ax.Databasename()));
info(StrFmt("ReplicaID: %1",ax.DatabaseReplicaId()));
```

Error codes

"2":	Database not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Error codes

"2":	Database not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

DatabaseName

DatabaseTitle

Syntax

```
DatabaseTitle()
```

Input

None

Output

(Text: Direct)
Title on current database.

Function

Returns the current database title, which should not be confused with the database name that specifies the file name of the database.
Can only be used after use of ***Logon***.

Example

```
ax2ln.Logon('help4.nsf');
print "Database file...: ",ax2ln.DatabaseName();
print "Database title..: ",ax2ln.DatabaseTitle();
pause;
```

Error codes

"2":	Database not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Error codes

"2":	Database not selected

Section

3.3.1

Other references

DatabaseName

Debug

Syntax

`Debug(<Debug>)`

Input

(Boolean)
Activation (TRUE) or deactivation (FALSE) of the debug-system.

Output

None

Function

Activates a function by which all calls to the DLL are shown in Windows. In addition, further information will often be shown if the DLL function is complex.

Example

```
ax2ln.Debug(TRUE);
ax2ln.Query('Form="Memo"');
ax2ln.Debug(FALSE);
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.10.2

Other references

SetLog

DeleteAttachAll

Syntax

```
DeleteAttachAll()
```

Input

None

Output

None

Function

Removes all attachments on the current document.
If the attachment is created by the user in a rich-text field, the icon of the attachment will not be deleted. The actual attachment will be deleted though.

Example

```
ax2ln.GetUnique("Number","123");
ax2ln.DeleteAttachAll(); //Remove old
ax2ln.AttachFile("c:\\autoexec.bat","Startup");
```

Error codes

"2":	Form not selected, use method ***Form***
"3":	Document not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.7

Other references

DetachFile
AttachFile

DeleteAttachment

Syntax

```
DeleteAttachment[<Attachment Name>]
```

Input

(Text)
Optional name on attachment.

Output

None

Function

Deletes an attachment in the current Notes document. Note that it is the attachment-name that is specified.
If the attachment is created by the user in a rich-text field, the icon of the attachment will not be deleted. The actual attachment will be deleted though.

Example

```
ax2ln.GetUnique('Number',"123");
ax2ln.DeleteAttachment('Startup'); // Remove old
ax2ln.AttachFile('c:\\autoexec.bat',"Startup");
```

Error codes

"1":	Attachment not found on current document
"2":	Form not selected, use method ***Form***
"3":	Document not selected
"98":	Attachment name is missing
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.7

Other references

DetachFile
AttachFile

DeleteCurrent

Syntax

```
DeleteCurrent()
```

Input

None

Output

None

Function

Deletes current document.

Example

```
ax2ln.Form('Department');
ax2ln.getUnique('Number',"125");
ax2ln.DeleteCurrent();
```

Error codes

"1":	No document selected
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.5.3

Other references

DeleteID
Commit

DeleteID

Syntax

```
DeleteID(<DocID>)
```

Input

(Text)
Unique DocId on Notes document to be deleted.

Output

None

Function

Deletes a document by selecting a document ID. Document ID may e.g. be retrieved by ***Query***, or by storing DocId in an AX table field.

Example

```
while  select  CustTable
where (CustTable.DeleteMark == 1)
{
    ax2ln.DeleteID(CustTable.NotesDocID);
    CustTable.Delete();
}
```

Error codes

"1":	Error in DocId
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.5.3

Other references

SelectID
DeleteCurrent
Commit

DeleteTmpFile

Syntax

```
DeleteTmpFile()
```

Input

None

Output

None

Function

Deletes temporary files used in connection with e.g. ***Query***.

Example

```
...
ax2ln.SetTmpFileName();
ax2ln.query('');
...
ax2ln.DeleteTmpFile();
```

Error codes

"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Error codes

None

Section

3.3.3

Other references

SetTmpFileName
GetTmpFileName

DetachFile

Syntax

```
DetachFile(<Filename>[,<Attachment Name>])
```

Input

(Text, Text)

1. Parameter: Filename on file to be detached to
2. Parameter: Optional name of attachment to be detached. If no parameter is supplied, an attachment with the name of the first Parameter is detached.

Output

None

Function

Detaches an attachment specified in the second Parameter of the current document. If a second Parameter is not specified, the attachment is selected with the same name as the file name specified in the first Parameter.
Attachment is detached to the file specified in the first Parameter.

Example

```
ax2ln.AttachFile('c:\\original',"Temp");
ax2ln.DetachFile('c:\\copy_',"Temp");
ax2ln.DeleteAttachment("Temp");
```

Error codes

"1":	Attachment not found on current document
"2":	Form not selected, use method ***Form***
"3":	Document not selected
"98":	Attachment name is missing
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.7

Other references

AttachFile
DeleteAttachment

DLLVersion

Syntax

```
DLLVersion()
```

Input

None

Output

(Text: Direct)
Version number and release date – e.g. "1.0.1.34 - 01.05.2003"

Function

Returns the current version number and release date of the Windows program library (ERP2LN.DLL).

Example

```
print "DLL version: ",ax2ln.DLLVersion();
pause;
```

Section

3.3.2

DocID

Syntax

```
DocID()
```

Input

None

Output

(Text: Direct)
The Document ID on the current active document in Notes.

Function

Returns the DocId of the active document in Notes. A document can be made active by ***GetUnique***, ***SelectID*** and ***QueryUnique*** (among others).
Can be used for checking whether a document is active before it is processed.

Example

```
if (StrLen(ax2ln.DocID())>2)
    ax2ln.SetFieldValue('Number',"1234");
```

Error codes

”0”;	Document with this DocId not found
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.5

Other references

UNID

ExistsAttach

Syntax

```
ExistsAttach(<Attachment name>)
```

Input

(Text)
The name on attachment to be looked for in the current document.

Output

(Boolean: Direct)

FALSE:	Either an attachment with that name exists on the current Document or an error has occurred (check with ***GetNotesText***)
TRUE:	Attachment is present on current document

Function

Is used to check if an attachment with the specified name exists in the selected document.

Example

```
ax2ln.SelectID("000013E0");
if (ax2ln.ExistsAttach('config.sys'))
    ax2ln.DetachFile('config.sys');
```

Error codes

"2":	Form not selected, use method ***Form***
"3":	No document selected
"98":	Attachment name is missing
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.7

Other references

AttachFile
DetachFile
DeleteAttachment

ExistsDatabase

Syntax

```
ExistsDatabase(<full path to Lotus Notes Database>)
```

Input

(Text)
Path and name on database to be checked for existence.

Output

(Boolean: Direct)

FALSE:	Database can not be found on slected server (or an error has occurred - use ***GetNotesText***)
TRUE:	Database exists on server.

Funktion

Used to check if a database is present on the selected server.

Eksempel

```
ax2ln.Init();
ax2ln.OpenServer('acme/com');
if(ax2ln.ExistsDatabase('customer\custtable.nsf')) {
  // Database is found
  ax2ln.Logon('customer\custtable.nsf');
  ..
} else {
  // Database does not exist!
  ..
}
```

Fejlkoder

"2":	Server not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Afsnit

3.9.1

Se også

OpenServer
Logon
ExistsField
ExistsForm
ExistsView

ExistsField

Syntax

```
ExistsField (<Fieldname>)
```

Input

(Text)
Name on Notes field to be checked for existence.

Output

(Boolean: Direct)

FALSE:	Either the field with that name exists on the current Form or an error has occurred (check with ***GetNotesText***)
TRUE:	Field is present on current form

Function

Is used for checking whether a field with the specified name exists in the current form. The field need not necessarily exist in the current document, only in the design of the form. Thus selection of a current document is not required to be able to use the method. However, it must be specified which form is being used (method ***Form***).

Example

```
ax2ln.SelectID("000013E0");
if (ax2ln.ExistsField('Number'))
    ax2ln.SetFieldValue('Number',"22");
```

Error codes

"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.1

Other references

ExistsForm
ExistsView
FirstField

ExistsForm

Syntax

```
ExistsForm(<Form name>)
```

Input

(Text)
Name on Notes form to be checked for existence. Aliases for form can also be used.

Output

(Boolean: Direct)

FALSE:	Either the form with that name or aliases exists in the current Database or an error has occurred (check with ***GetNotesText***)
TRUE:	Form is present in the current database

Function

Is used for checking whether a form with the specified name exists in the current database. Selection of database must take place before the method is called (use ***Logon***).
If the form is defined with aliases, calling this method with one of these aliases will also re-turn TRUE.

Example

```
if (ax2ln.ExistsForm('Demo'))
    ax2ln.Form('Demo');
else
    print "The form 'Demo' is not available";
pause;
```

Error codes

"2":	No logon to database performed
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.1

Other references

ExistsField
ExistsView
FirstForm

ExistsView

Syntax

```
ExistsView(<View name>)
```

Input

(Text)
Name on Notes form to be checked for existence. Aliases for view can also be used.

Output

(Boolean: Direct)

FALSE:	Either the view with that name or alias exists in the current Database or an error has occurred (check with ***GetNotesText***)
TRUE:	View is present in the current database

Function

Is used for checking whether a view with the specified name exists in the current database. Selection of database must take place before the method is called (use ***Logon***).

Example

```
if (ax2ln.ExistsView('All'))
    ax2ln.View("All");
else
    print "The view 'All' must exist in Notes";
pause;
```

Error codes

"2":	No logon to database performed
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.1

Other references

ExistsField
ExistsForm
FirstView

FieldType

Syntax

```
FieldType(<Fieldname>)
```

Input

(*Text*)
Specification of fieldname

Output

(Text: Direct)
Type of field in Notes. Could also specify an error code.

Function

Is used to read the field-type the field is in Lotus Notes, e.g. Rich-text or Number. For a complete list of field types that can be returned, see Appendix 4
If no Parameters are specified, current field is used.
Form must be selected by ***Form*** before use of the method.

Example

```
ax2ln.Form('Test');
ax2ln.GetUnique('Nummer',"SALES");
if (ax2ln.FieldType('Date')=='DateTime')
    ax2ln.SetFieldValue('Date',
      Date2Str(Today(),123,2,3,2,3,2));
```

Error codes

"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.3

Other references

FormType

FirstAttach

Syntax

```
FirstAttach()
```

Input

None

Output

(Text: Direct)
Name of the first attachment on the current document.

Function

Points to the first attachment attached to the current document and returns the name of this attachment.
This method is useful in combination with ***NextAttach*** when detaching all the attachments on a document.

Example

```
attach=ax2ln.FirstAttach();
while (StrLen(attach)>2)
{
    print 'Found attach: ',attach;
    ax2ln.DetachFile('c:\\temp\\'+attach,attach);
    attach=ax2ln.NextAttach();
}
```

Error codes

"":	No attachments on current document
"2":	Form not selected, use method ***Form***
"3":	No document selected
"98":	Attachment name is missing
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.9.3

Other references

NextAttach

FirstField

Syntax

```
FirstField()
```

Input

None

Output

(Text: Direct)
Name of the first field on the current form design.

Function

Points to the first field defined in the form (not the current form but the design) and return the name of the field.
This method is useful in combination with ***NextField*** when reading the fieldnames on a Lotus Notes form.

Example

```
f = ax2ln.FirstField();
while (f != '')
{
    print f;
    F = ax2ln.NextField();
}
```

Error codes

"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

NextField
FirstForm
FirstView

FirstForm

Syntax

```
FirstForm()
```

Input

None

Output

(Text: Direct)
Name of the first form in the current selected database.

Function

Selects the first form in the current database and returns the name of the form.
Names of both ordinary forms and subforms are returned.
Before use, database must have been selected by ***Logon***.

Example

```
f = ax2ln.FirstForm();
while (f != '')
{
    print f;
    f = ax2ln.NextForm();
}
```

Error codes

"2":	No logon to database performed.
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

NextForm
FirstField
FirstView

FirstView

Syntax

```
FirstView()
```

Input

None

Output

(Text: Direct)
Name of the first view in the current selected database.

Function

Selects the first view in the current database and returns the name of this view.
Before use, database must have been selected by ***Logon***.

Example

```
v = ax2ln.FirstView();
while (f != '')
{
    print v;
    v = ax2ln.NextView();
}
```

Error codes

"2":	No logon to database performed.
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

NextView
FirstField
FirstForm

Form

Syntax

```
Form(<Form name>)
```

Input

(Text)
Name of form in Lotus Notes to use. Aliases can be used.

Output

None

Function

Specification of form to be used for operations in Notes – e.g. data-validation. It is also possible to use form aliases.

Example

```
ax2ln.Form('Person');
ax2ln.QueryUnique("Name='John Williamson ");
```

Error codes

"1":	Form not found in current database.
"9":	Logon to database is missing
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.3

Other references

FormName

FormName

Syntax

```
FormName()
```

Input

None

Output

(Text: Direct)
Name on current form in Notes.

Function

Is used to read which form is open in the current session of Lotus Notes.

Example

```
if (ax2ln.FormName() != 'Department')
    print "Form not correct";
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.3.1

Other references

FormName

FormType

Syntax

```
FormType()
```

Input

None

Output

(Text: Direct)
Type of form on current form. Could also return an error code.

Function

Is used to read of which type the current form is in Notes.
The following types can be returned:
"Main"
"Response to Main"
"Response to Response"
Form must have been selected by the method ***Form*** before use of this method.
Note that any error codes may be returned. It is suggested that the length of return value is tested.

Example

```
ax2ln.Form('Test');
print "Formtype: ",ax2ln.FormType();
```

Error codes

"2":	Form not selected, use method ***Form***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.9.3

Other references

Form

GetDialogDatabaseName

Syntax

```
GetDialogDatabaseName()
```

Input

None

Output

(Text: Direct)
Database selected at last call to ***OpenDialog***.

Function

Returns the database name of the database selected at last call to ***OpenDialog***. Can be used, when ***OpenDialog*** is not called with the OPEN parameter to examine the selected data.

Example

```
ax2ln.OpenDialog();   // Used to select DB
ax2ln.OpenServer('DefaultServer/Acme');
ax2ln.Logon(ax2ln.GetDialogDatabaseName());
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.1

Other references

OpenDialog

GetDialogServerName

Syntax

```
GetDialogServerName()
```

Input

None

Output

(Text: Direct)
Server selected at last call to ***OpenDialog***.

Function

Returns the server name of the database selected at last call to ***OpenDialog***. Can be used, when ***OpenDialog*** is not called with the OPEN parameter to examine the selected data.

Example

```
ax2ln.OpenDialog();  // Used to select server
ax2ln.OpenServer(ax2ln.GetDialogServerName());
ax2ln.Logon(FixedDatabase);
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.1

Other references

OpenDialog

GetDocInfo

Syntax

```
GetDocInfo(<PROPERTY>)
```

Input

(Text)
The type of information (property) wanted

Output

(String: Direct)
Property value

Function

Is used for searching for information about the current document. The following information is available:

"DOCID"	The DocId of the current document
"UNID"	UNID
"ATTACH_COUNT"	The number of attachments
"CREATED"	The creation date
"LAST_MODIFIED"	The last modification
"LAST_ACCESSED"	The last time the document was accessed
"ADDED"	Added to this replica
"IS_RESPONSE"	Response document ("0": No,"1": Yes)

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Example

```
ax2ln.SearchFirst();
print "Last modified: ",ax2ln.GetDocInfo('LAST_MODIFIED');
pause;
```

Section

3.5.5

Other references

GetItemInfo

GetFieldLength

Syntax

```
GetFieldLength(<Fieldname>)
```

Input

(*Text*)
Notes field name.

Output

(Integer: Direct)
>=0: Number of characters in field
<0: Error, see below

Function

Returns the size of data in supplied field. It is the length in characters returned – not the size in bytes.

Error codes

”-1”:	Field supplied or not found on current form
”-2”:	Form not selected
”-3”:	Field length exceeded
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.5

Other references

GetItemInfo

GetFieldValue

Syntax

```
GetFieldValue(<Fieldname>)
```

Input

(Text)
Specification of the name of the item in Notes whose value should be read.

Output

(Text: Direct)
Item value

Function

Query for item value in the current document. The document must be selected before the query. Unlike what happens when ***QueryField*** is used, it will not be checked if the item also exists as a field on the current form.

Example

```
ax2ln.GetUnique("Number","123");
print "Name   : ",ax2ln.GetFieldValue("Name");
print "Ref    : ",ax2ln.GetFieldValue("Ref");
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.5

Other references

PutFieldValue
QueryField

GetFoundDocs

Syntax

```
GetFoundDocs()
```

Input

None

Output

(Number: Direct)
Number of documents found in last search (***Query***)

Function

Returns the number of documents found at last call to ***Query***.

Example

```
ax2ln.Query("@all");
print "Found documents: ", ax2ln.GetFoundDocs();
pause;
```

Error codes

-2:	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.2

Other references

Query

GetHandle

Syntax

```
GetHandle()
```

Input

None

Output

(Number)
The handle number of the current session.

Function

Returns the handle number of the current session. This is relevant if ***SetHandle*** is used to switch between sessions.

Example

```
if (ax2ln.GetHandle() != 2)
{
    ax2ln.SetHandle(2);
    ax2ln.OpenServer("Local");
    ax2ln.Logon("MyDB.nsf");
    ax2ln.Form("Form 2");
    ax2ln.View("View 2");
}
```

Error codes

"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.2.4

Other references

SetHandle

GetItemInfo

Syntax

```
GetItemInfo()
```

Input

(Text, Text)
1st parameter: The name of the Notes field in which to search
2nd parameter: Property searched for

Output

(String: Direct)
Property value

Function

Used for searching for information about a specific item (specific data) in the current document. The following information is available:

"TYPE" :	Data type (TEXT, NUMBER, DATETIME...)
"LENGTH" :	Length of the data in bytes

Error codes

”1”	Field is not defined on Notes form
”2”:	Form not selected, use method ***Form***
”3”:	Internal format-error (field and flag must be separated with ”@@@”)

Section

3.5.5

Other references

GetDocInfo

GetLink

Syntax

```
GetLink()
```

Input

None

Output

(Text: Direct)
Link to current document.

Function

This method returns a notes-link to the current selected document in Lotus Notes. This link is unique across databases, servers and replicas.

The link is build like this:

“notes://<server>/<database_replicaid/<view_unid>/<document unid>”

Example

```
ax2ln.SearchFirst();
print "link: ",ax2ln.GetLink();
```

Error codes

“”:	Server, view or document not selected.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Other references

MailGetLink

GetMessages

Syntax

```
GetMessages()
```

Input

None

Output

(String: Direct)

FALSE:	Error messages will not be shown.
TRUE:	Error messages will be shown.

Function

Returns status specifying whether error messages are shown in separate window's when an error occurs when integrating AX and Notes.
See ***Messages*** for a more detailed description.

Section

3.10.1

Other references

Messages

GetNotesText

Syntax

```
GetNotesText()
```

Input

None

Output

(Text: Direct)
Returns the resultcode/errorcode from last call to DLL.

Function

After each call to the DLL a result-code is returned to AX – including error-codes. This method returns this value.
Use ***GetNotesText*** to implement error-handling.

Example

```
ax2ln.form('Test');
if (ax2ln.GetNotesText() != '')
{
    print "Error opening 'Test'";
    pause;
    return;
}
```

Section

-

Other references

-

GetRichSepMode

Syntax

```
GetRichSepMode()
```

Input

None

Output

(String: Direct)

FALSE:	Rich-text fields is stored in main comma file
TRUE:	Rich-text fields is stored in separate comma files

Function

Returns status specifying how rich-text files are returned upon ***Query***. See under ***SetRichSepMode*** and section 3.6.2.1

Section

3.6.2.1

Other references

SetRichSepMode

GetTmpFileName

Syntax

```
GetTmpFileName()
```

Input

None

Output

(String: Direct)
Name of the current temporary file.

Function

Returns the name of the current temporary file.

Section

3.3.3

Other references

SetTmpFileName

GetUnique

Syntax

```
GetUnique(<Fieldname>,<Fieldvalue>)
```

Input

(Text, Text)
1. Parameter: Name on lookup field in Notes.
2. Parameter: Lookup value to be searched for.

Output

(Text)
DocId on the found document.
If return value is "99" or less, then this specifies an error code.

Function

This method is used for finding a document in Notes via a key. The key field is specified as first Parameter. A query is made in Lotus Notes for a document that has the second Parameter value in its field.
The method returns an error if the query finds more documents.
If there is a possibility that the query will result in the finding of more documents, the method ***Query*** should be used.
Remember that Notes is case-sensitive as regards field names.

Example

```
res = ax2ln.GetUnique('Number',Number);
if (strLen(res) > 2)  // Found
{
    print "Notes Ref: ", ax2ln.QueryField("Ref");
    pause;
}
```

Error codes

"0":	Document not found
"1":	Field is not defined on Notes form
"2":	Form not selected, use method ***Form***
"3":	More than 1 document found
"4":	Wrong number of arguments (should be 2)
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.6

Other references

SelectID
Query
QueryUnique

ImportFile

Syntax

```
ImportFile(<Fieldname>,<Filename>)
```

Input

(Text, Text)

1. Parameter:	Name for rich-text field
2. Parameter:	Filename to imported from

Output

None

Function

Imports the file specified in the second Parameter to the rich-text field specified in the first Parameter. Is used for embedding graphics and rich-text. Can be combined with ***SetRichFieldValue*** for "ordinary" body text.
The specified file is added to the existing contents in the rich-text field.
Note that some of the graphics formats supported have some variants that are not all supported by the integration kit. It is recommended to test the compatibility of existing graphics programs by sending a mail to one's own address.

Example

```
ax2ln.SetRichFieldValue('Body','Rich text fil:');
ax2ln.ImportFile('Body','c:\\temp\\fil.rtf');
ax2ln.SetRichFieldValue('body','------------');
ax2ln.Commit();
```

Error codes

"1":	Field is not defined on Notes form
"2":	Form not selected, use method ***Form***
"3":	The field is not a rich-text field
”4”:	Current document not selected
"98":	File not found or cannot be read.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.2

Other references

SetRichFieldValue
MailBodyFile

Init

Syntax

```
Init[(<DLL fil>)]
```

Input

(Text)
Optional filename on DLL file.

Output

None

Function

Creation of link to the Windows program library that handles communication to Lotus Notes.
This method must be called to be able to use the integration methods.
It is possible to specify another name of the DLL program library. If no Parameter is specified, the default value "ERP2LN.DLL" is used.

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.1

Other references

DLLVersion

IsDemoVersion

Syntax

```
IsDemoVersion()
```

Input

None

Output

(Boolean: Direct)

FALSE:	NOT a demo version of DLL
TRUE:	DLL IS a demo version

Function

Tells whether the current integration DLL is a demo version (not slized with production serial number).

Section

3.3.2

Other references

DLLVersion
Show

Logon

Syntax

```
Logon(<Database name>[,<Force logon>])
```

Input

(Text [,Boolean])

1. Parameter:	Filename on Notes database.
2. Parameter:	Optional specification that logon should be per formed even if a logon to current database has been carried out.

Output

None

Function

Opens a session with Lotus Notes upon specification of a database name. On login Lotus Notes checks the rights of the user. The current set-up of the workstation (user ID, location etc.) is used when creating the link. When contact to a Notes server is established for the first time, the user's password is requested – unless the Notes client has been set up to share password with other programs.
If the optional extra parameter (TRUE) is supplied, the logon is performed even if there already is an open connection to the database.
If this parameter is not supplied logon is only performed if the current database is different from the supplied database.

Example

```
ax2ln.Init();
ax2ln.openServer('Local');
ax2ln.Logon('Department');
if (ax2ln.GetNotesText() != '')
    print 'Error connecting to Department on '+
          'local client';
```

Error codes

"9":	Logon not completed because server name is missing. Use method OpenServer to supply server name.
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.2

Other references

DatabaseName

LookupName

Syntax

```
LookupName(Server,Views,Names,Items)
```

Input

(Text)

1. Parameter:	Server where names.nsf is located.
2. Parameter:	List of names to search for.
3. Parameter:	views to search in address book.
4. Parameter:	Items to return.

Output

Itemsvalues for found document

Function

Make a lookup in e.g. the name and address book for a specific name.
Returns information from the fieldnames given in parameter 4.
Use '' as servername (parameter 1) is blank ('') the local PC will be used.
Normally '' is used as view (parameter 2) as the default lookup view will then be used ($Users).
Serveral names can be supplied as parameter 3. They must be seperated by @@@
(e.g. 'john@@@johnny').
Similar several items can be returned. Also use @@@ as delimiter.

If more documents are found, they will be returned seperated by @@@. The individual items returned are seperated by ###.
The function always return the used view as first item and the matchup value as the second part.

Example

```
ax2ln.openServer('Local');
Print ax2ln.LookupName('','','John', 'FullName@@@MailAddress')
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

MailActiveDocument

Syntax

```
MailActiveDocument()
```

Input

None

Output

(Text)
Document UNID on active mail-document.

Function

Returns UNID on the document the user has currently open in the Lotus Notes client. If the Lotus Notes client is not open or the user has no open document an empty string is returned.
This methods differs from ***UNID***, in that ***UNID*** returns the current selected document by the integration kit (AX), not (necessarily) the document open in the Lotus Notes client.

Example

```
ax2ln.MailSetSendTo("johndoe@acme.com");
ax2ln.MailSetSubject('Dear John');
ax2ln.MailComposeOLE(FALSE);
print ax2ln.MailActiveDocument();
```

Error codes

"":	No active document

Other references

MailComposeOLE
UNID

MailAttachFile

Syntax

```
MailAttachFile(<Filename>[,<AttachmentName>])
```

Input

(Text, Text)

1. Parameter:	Filename on file to be attached current mail.
2. Parameter:	Optional name of attachment on mail.

Output

None

Function

Attaches the file specified in the first Parameter of the current mail *(note that the file is* <u>*not*</u> *attached to a specific rich-text field, only to the actual document).*
The attachment will appear at the bottom of the Notes document with its file name. If another Parameter is specified, the attachment will get this name in the Notes mail. This option may e.g. be used for a more descriptive name.

Example

```
ax2ln.MailPrepare('nn@acme.com',"Testmail");
ax2ln.MailSetBodyText(Have a look');
ax2ln.MailAttachFile('c:\\joke.bmp','Funny');
ax2ln.MailSend ();
```

Error codes

"2":	Form not selected, use method ***Form***
”3”:	No document selected.
"98":	File not found or cannot be read.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.8.2

Other references

AttachFile
MailBodyFile

MailComposeOLE

Syntax

```
MailComposeOLE([<Backend dokument>])
```

Input

([Boolean])
Optional indication whether a backend document should be created.

FALSE:	Do NOT create a new document (default)
TRUE:	Create a new document

Output

(Text)
Document UNID on mail-document or error code.

Function

Opens a new mail in the Lotus Notes mail client with the data set from the other mail-methods (e.g. ***MailPrepare***).
If TRUE is supplied as parameter, a new document will be created in Lotus Notes. This document can then be modified with the methods in the integration kit.
If no parameter is supplied (or FALSE is used), no document is created in the mail database, only a "front-end" mail will be prepared.
If one or more attachments have been created on the current mail, a back-end document is always created (to store the attachments).

This method returns to AX instantly – there will be no check whether the mail is sent or not. If a backend document has been created the UNID is returned.

Example

```
ax2ln.MailPrepare('nn@acme.com',"Testmail");
ax2ln.MailSetBodyText('AX mail');
ax2ln.MailComposeOLE();
```

Error codes

"":	Mail document not found or created
"1":	Error activating mail-client
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.8.4

Other references

MailPrepare

MailCreateLinkFile

Syntax

```
MailCreateLinkFile()
```

Input

(Text)

1. Parameter:	Filename on file to be created with Lotus Notes link-information (must have the extension '.url').

Output

None

Function

Works as ***CreateLinkFile***, although with this method the file contains a link to a sent mail. The method is typically called right after ***MailComposeOLE*** or ***MailSend***.

Other references

CreateLinkFile
MailGetLink
MailComposeOLE
MailSend

MailGetDB

Syntax

```
MailGetDB()
```

Input

None

Output

(Text)
Path to mail database, as entered in notes.ini.

Function

Returns the name of the current mail database. This information is fetched in notes.ini.

Example

```
ax2ln.init();
print ax2ln.MailGetDB();
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

MailSetDB

MailGetHandle

Syntax

```
MailGetHandle()
```

Input

None

Output

(Integer: Direct)
Number of handle used for mails

Function

Several connections can be open at any time to Lotus Notes. A special handle is used for mail-connection. This handle is identical to the maximum number of open handles.
Normally this is 16

Error codes

Other references

GetHandle
SetHandle

MailGetLastUNID

Syntax

```
MailGetLastUNID()
```

Input

None

Output

(Text: Direct)
Notes Universal Document Id of the last sent mail-document in Notes.

Function

Returns the Notes Universal Document Id of the last sent mail document in Notes. The Notes Universal Document Id is a 32-character string which is unique in a database across replicas.

This function can be used, for journaling mails sent by e.g. batch processes.

Error codes

“”:	Server, view or document not selected.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Other references

UNID

MailGetLink

Syntax

```
MailGetLink()
```

Input

None

Output

(Text: Direct)
Link to current mail-document.

Function

This method returns a notes-link to the current prepared or sent mail in Lotus Notes. This link is unique across databases, servers and replicas.

The link is build like this:

“notes://<server>/<database_replicaid/<view_unid>/<document unid>”

Error codes

“”:	Server, view or document not selected.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Other references

GetLink
MailCreateLink

MailGetServer

Syntax

```
MailGetServer()
```

Input

None

Output

(Text, Direct)
Name of mail-server

Function

Returns the name of the server processing mails. Information is fetched from notes.ini

Example

```
ax2ln.init();
print ax2ln.MailGetServer();
```

Error codes

"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

MailSetServer

MailGetUsername

Syntax

```
MailGetUsername()
```

Input

None

Output

(Text, Direct)
The Lotus Notes user name for the current user.

Function

Returns a fully qualified Lotus Notes user name, e.g.: ”CN=John Smith/O=Intoint”.

Example

```
userName = ax2ln.MailGetUsername();
ax2ln.SetFieldValue("Principal",userName);
ax2ln.Commit();
```

MailPrepare

Syntax

```
MailPrepare(<SendTo>,<Subject>[,<CC>)[,<BCC>]])
```

Input

(Text, Text[, Text[, Text]])

1.	Parameter:	E-mail address of receiver(s).
2.	Parameter:	Subject on mail.
3.	Parameter:	Optional. CC(s) of mail.
4.	Parameter:	Optional. BCC(s) of mail.

Output

None

Function

Prepares the mail for sending. This method initializes the mail, and the contents are subsequently filled in with the methods ***MailBodyXXX***. Note that the third and fourth parameters are optional.
E-mail addresses (also CC and BCC) can be separated by commas to specify more receivers of the mail.

Example

```
ax2ln.MailPrepare ('nn@acme.com,nn@acme.com',
                   'Test','','John Doe/Acme');
ax2ln.MailSetBodyText("What’s up?");
ax2ln.MailSend();
```

Section

3.8.1, 3.8.2

Other references

MailBody
MailBodyText
MailBodyFile
MailSend

MailSend

Syntax

```
MailSend([<Filename>])
```

Input

(Text)
Optional filename to be imported into the body field in mail.

Output

None

Function

Sends prepared mail. If a file name is specified, the file is imported at the end of the body text field of the mail, even if this field has been originally built with the method ***MailSetBody*** or ***MailSetBodyText***.

Example

```
ax2ln.MailPrepare ('nn@acme.com','My config.sys');
ax2ln.MailSend('c:\\config.sys');
```

Error codes

"1":	Mail not prepared (e.g. SendTo missing)
"98":	File not found or cannot be read.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.8.3

Other references

MailPrepare
MailBodyFile

MailSendDocument

Syntax

```
MailSendDocument([<Filename>])
```

Input

None

Output

None

Function

Sends the current active document.
Any document in Lotus Notes can be send as mail. This method sends the current active docuemnt. Remember to either call MailSetSendTo or enter a value in a SendTo field on the document befor calling command.

Example

```
ax2ln.SelectID('123456');
ax2ln.MailSetSendTo('johndoe@acme.com');
ax2ln.MailSendDocument();
```

Error codes

"1":	Mail not prepared (e.g. SendTo missing)
"98":	File not found or cannot be read.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.8.3

Other references

MailSend

MailSendFrom

Syntax

```
MailSendFrom(<SendFrom>)
```

Input

(Text)
Specification of name of sender of mail (SendFrom).

Output

None

Function

Supplies mail-addresses to the sender of the mail.
If not set the mail will be stamped with the current users mail address.

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSendFrom('"john Doe" <jd@acme.com>');
ax2ln.MailSend();
```

Other references

MailPrepare

MailSetBCC

Syntax

```
MailSetBCC(<BCC>)
```

Input

(Text)
Specification of BCC on mail

Output

None

Function

Supplies mail-addresses to the BCC field on new mail
BCC can also be supplied as parameter when calling ***MailPrepare***

Example

```
ax2ln.MailPrepare ('nn@acme.com','Testmail');
ax2ln.MailSetBCC('abc@ef.gh');
ax2ln.MailSetBody('My mail');
ax2ln.MailSend ();
```

Section

3.8.1

Other references

MailPrepare
MailSetCC
MailSetSendTo

MailSetBody

Syntax

```
MailSetBody(<Text>)
```

Input

(Text)
Body-text for mail

Output

None

Function

Adds text to mail in the body field. By repeated calls of this method, more lines are built up.
*This method **__must__ be used when importing other file formats with the method Mail-BodyFile**.*
Corresponds to ***MailBodyText*** with regard to functions but is slightly slower as all text is transferred via temporary file.

Example

```
ax2ln.MailPrepare ('nn@acme.com','Testmail');
ax2ln.MailSetBody('First line');
ax2ln.MailSetBody('Second line');
ax2ln.MailSend();
```

Error codes

”1”:	Error opening mail-server and/or mail-database
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.8.2

Other references

MailBodyText
MailBodyFile

MailSetBodyFile

Syntax

```
MailSetBodyFile(<Filename>)
```

Input

(Text)
Name of file to be imported in to current mail.

Output

None

Function

Imports file to the current mail. Is used for embedding graphics and rich-text. Is combined with ***MailBody*** for "ordinary" body text.
See Section 3.8.2 for more detailed information about use of this method.
Note that some of the graphics formats supported have some variants that are not all supported by the integration kit. It is recommended to test the compatibility of existing graphics programs sending a mail to one's own address.

Example

```
ax2ln.MailPrepare('nn@acme.com','Testmail');
ax2ln.MailSetBody('Rich text file:');
ax2ln.MailSetBodyFile('c:\\temp\\fil.rtf');
ax2ln.MailSetBodyFile('c:\\temp\\sign.pcx');
ax2ln.MailSend();
```

Error codes

"98":	Error writing to temporary file
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.8.2

Other references

MailBody
MailSend

MailSetBodyText

Syntax

```
MailBodyText(<Text>)
```

Input

(Text)
Body text for mail

Output

None

Function

Adds text to mail in the body field. By repeated calls of the method, more lines are built up.
This method is ***used when importing of other file formats with the method Mail-BodyFile is NOT wanted.***
Corresponds to ***MailBodyText*** with regard to functions but is quicker as all transfer to Notes happens directly.

Example

```
ax2ln.MailPrepare("nn@acme.com","Testmail");
ax2ln.MailSetBodyText("First line");
ax2ln.MailSetBodyText("Second line");
ax2ln.MailSend();
```

Error codes

"1":	Can not be combined with ***MailBody*** or ***MailBodyFile*** in the same mail
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.8.2

Other references

MailBodyText
MailBodyFile
MailSend

MailSetCC

Syntax

```
MailSetCC(<CC>)
```

Input

(Text)
Specification of CC on new mail

Output

None

Function

Supplies mail-addresses to the CC field on new mail.
CC can also be supplied as parameter when calling ***MailPrepare***

Example

```
ax2ln.MailPrepare('nn@acme.com','Testmail');
ax2ln.MailSetCC('abc@ef.gh');
ax2ln.MailSetBody('My mail');
ax2ln.MailSend();
```

Section

3.8.1

Other references

MailPrepare
MailSetBCC
MailSetSendTo

MailSetCopyDB

Syntax

```
MailSetCopyDB(<CC>)
```

Input

(Text)
Maildatabase to copy outgoing mails to

Output

None

Function

Defines a database, where a copy of outgoing mails should be copied to, when sending mails.

Example

```
ax2ln.init();
ax2ln.MailSetCopyDB('mail\\CommonMails.nsf');
ax2ln.MailPrepare('nn@acme.com', 'Test mail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSend();
```

Other references

CopyToDatabase
MailSetCopyToUser

MailSetCopyToUser

Syntax

```
MailSetCopyToUser (<doCopy>)
```

Input

(Boolean)

FALSE:	Don’t perform any copying
TRUE:	Copy outgoing mails to current users mailbox

Output

None

Function

If outgoing mails has been set up to be send though a different mail-database, using the method MailSetDB, this command can specify, that a copy of the outgoing mail should also be copied to the current users normal mail-database.
Same as MailSetCopyDB, but with no need to specify the database, as the current users database are known to the system.

Example

```
ax2ln.MailPrepare('nn@acme.com','Testmail from other mail DB');
ax2ln. MailSetDB('anothermailDB.nsf');
ax2ln.MailSendFrom('sales@intoint.com');
ax2ln.MailSetCopyToUser(true);
ax2ln.MailSend();
```

Other references

MailSetDB
MailSetCopyDB

MailSetDB

Syntax

```
MailSetDB(<mail-database>)
```

Input

(Text)
1. parameter: Name of mail-database

Output

None

Function

Enables the temporary selection of an alternative mail-database. This selection will not be entered into notes.ini.

Example

```
ax2ln.init();
ax2ln.MailSetDB('mail\alternative.nsf');
ax2ln.MailSendFrom('sales@intoint.com');
ax2ln.MailSetReplyTo('info@intoint.com');
ax2ln.MailPrepare('nn@acme.com', 'Regarding…');
ax2ln.MailSend();
```

Other references

MailGetDB

MailSetDeliveryPriority

Syntax

```
MailSetDeliveryPriority (<Priority>)
```

Input

(Number)
Specification of how soon Notes should route the mail

Output

None

Function

Corresponds to the “Delivery Priority” field on the Delivery Options in the Lotus Notes mail application.
Possible values:

1:	Low (Internally stores as “L”)
2:	Medium (Internally stores as “M”)
3:	High (Internally stores as “H”)

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetDeliveryPriority(3);
ax2ln.MailSend();
```

Other references

MailSetImportance

MailSetDeliveryReport

Syntax

```
MailSetDeliveryReport(<Report level>)
```

Input

(Text)
Specification of which kinds of reports regarding mail-progress should be returned

Output

None

Function

Corresponds to the “Delivery Report” field on the Delivery Options in the Lotus Notes mail application.
Possible values:

“N”:	None – no reports send – even if errors occur
“B”:	Only on failure - only report if Notes cannot deliver the message
“C”:	Confirm delivery – report when delivered (or not delivered)
“T”:	Trace entire path – report from every mail-server mail passes.

The parameter value corresponds to how Lotus Notes internally saves this item.

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetDeliveryReport(“C”);
ax2ln.MailSend();
```

Other references

MailSetImportance
MailSetReturnReceipt

MailSetImportance

Syntax

```
MailSetImportance(<Importance-level>)
```

Input

(Number)
Importance of mail – from low to high (1-3)

Output

None

Function

Corresponds to the “Importance” field on the Delivery Options in the Lotus Notes mail application. Setting this value to 3 (High) will result in an exclamation point next to the message in the Inbox of recipients
Possible values:

1:	High
2:	Medium (Default)
3:	Low

Not to be mixed up with MailSetDeliveryPriority that determines how fast the mail will be sent.

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetImportance(3);
ax2ln.MailSend();
```

Other references

MailSetDeliveryPriority

MailSetError

Syntax

```
MailSetError(<E-mail recepient>)
```

Input

(Text)
E-mail address on person to receive all DLL error messages.

Output

None

Function

It is possible to forward any error-message from the integration kit to one or more mail receivers. This is instantiated by calling this method.
Any error-message from the kit is then also sent as an e-mail.
This can ease support on systems running unattended and minimize down-time.

Example

```
ax2ln.Init();
ax2ln.MailSetError('John Doe/Acme');
```

Section

3.8.6

Other references

Messages

MailSetEncrypt

Syntax

```
MailSetEncrypt(<Do encrypt>)
```

Input

(Boolean)

Output

None

Function

Corresponds to the “Encrypt” field on the Delivery Options in the Lotus Notes mail application.
Specifies if mail should be encrypted (TRUE)

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetEncrypt(TRUE);
ax2ln.MailSend();
```

Other references

MailSetMAPI

Syntax

```
MailSetMAPI(<MAPI>)
```

Input

(Boolean)

FALSE:	Use Lotus Notes as e-mail system
TRUE:	Use MAPI for e-mails

Output

None

Function

Specification of whether to use MAPI as mail-protocol or Lotus Notes. Use parameter TRUE to switch to MAPI protocol.

Example

```
ax2ln.MailSetMAPI(True);
ax2ln.MailPrepare('intoint@intoint.com','Hello');
ax2ln.MailSend();
```

Section

3.8.5

Other references

MailSetMAPIPassword
MailSetMAPIProfile

MailSetMAPIPassword

Syntax

```
MailMAPIPassword(<MAPI Password>)
```

Input

(Text)
Password to be used when logging in to MAPI

Output

None

Function

If the MAPI setup requires password when using MAPI, the password can be send through this command.

Example

```
ax2ln.MailSetMAPI(TRUE);
ax2ln.MailSetMAPIProfile("MS Exchange Settings");
ax2ln.MailSetMAPIPassword("MyPassword");
```

Section

3.8.5

Other references

MailSetMAPI
MailSetMAPIProfile

MailSetMAPIProfile

Syntax

```
MailMAPIProfile(<MAPI Profile>)
```

Input

(Text)
Profile to be used when logging in to MAPI.

Output

None

Function

To avoid having to manually select a profile when starting e.g. MS Outlook, the profile can be specified by ***MailSetMAPIProfile***.

Example

```
ax2ln.MailSetMAPI(TRUE);
ax2ln.MailSetMAPIProfile("MS Exchange Settings");
ax2ln.MailSetMAPIPassword("MyPassword");
```

Section

3.8.5

Other references

MailSetMAPI
MailSetMAPIPassword

MailSetReplyTo

Syntax

```
MailSetReplyTo(<ReplyTo>)
```

Input

(Text)
Specification of mailaddress where mails should be replied to.

Output

None

Function

Supplies ReplyTo-addresses to the sender of the mail.
If not set the mail will be stamped with the current users mail address.

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetReplyTo('"john Doe" <jd@acme.com>');
ax2ln.MailSend();
```

Other references

MailPrepare
MailSendFrom

MailSetReturnReceipt

Syntax

```
MailSetReturnReceipt(<Do send returnn receipt>)
```

Input

(Boolean)
TRUE = Receive message when mail has been opened

Output

None

Function

Corresponds to the “Return Receipt” field on the Delivery Options in the Lotus Notes mail application.
Possible values:

FALSE:	Do <u>not</u> return a receipt (default)
TRUE:	Do return a receipt when mail has been opened by recipient

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSetReturnReceipt(TRUE);
ax2ln.MailSend();
```

Other references

NotesMailSetDeliveryReport

MailSetSave

Syntax

```
MailSetSave(<SAVE>)
```

Input

(Boolean)

FALSE:	Do not save sent sent mail-documents
TRUE:	Save sent mail-documents in "Sent"

Output

None

Function

Whether sent mails are saved or not are determined by the current user's Lotus Notes set-up. With MailSetSave it is possible to manually decide whether sent mails should be saved or not.

Example

```
ax2ln.init();
ax2ln.MailSetSave(TRUE);
ax2ln.MailPrepare('nn@acme.com', 'Regarding…');
ax2ln.MailSend();
```

Other references

MailSetSendTo

Syntax

```
MailSetSendTo(<SendTo>)
```

Input

(Text)
Specification of receiver of mail (SendTo).

Output

None

Function

Supplies mail-addresses to the SendTo field on new mail. SendTo can also be supplied as first parameter when calling ***MailPrepare***

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSend();
```

Section

3.8.1

Other references

MailPrepare

MailSetServer

Syntax

```
MailSetServer(<Server name>)
```

Input

(Text)
1. parameter: Name of mail-server.

Output

None

Function

This method enables temporary selection of an alternative mail-server. This selection will not be entered into notes.ini. At next use of the mail-server from notes.ini will be used.

Example

```
ax2ln.init();
ax2ln.MailSetServer('new_server/domain');
ax2ln.MailPrepare('nn@acme.com', 'Regarding …');
ax2ln.MailSend();
```

Other references

MailGetServer

MailSetSubject

Syntax

```
MailSetSubject(<Text>)
```

Input

(Text)
Specification of subject on new mail.

Output

None

Function

Supplies subject to the new mail. Subject can also be supplied as second parameter when calling ***MailPrepare***

Example

```
ax2ln.MailSetSendTo('nn@acme.com');
ax2ln.MailSetSubject('Testmail');
ax2ln.MailSetBody('My mail');
ax2ln.MailSend();
```

Section

3.8.1

Other references

MailPrepare

Messages

Syntax

```
Messages(<ShowMessages>)
```

Input

(Text)
Specification of error-messages should be shown (TRUE) or not (FALSE).

Output

None

Function

This method makes it possible to "hide" error messages. This may be useful in connection with e.g. batch updates etc. However, it requires that efficient test functions are built into the AX code that read ***GetNotesText*** upon each method and reacts on error situations.
Parameter FALSE de-activates error messages. Parameter TRUE re-activates error messages.
*Note that these settings are global to the entire AX session **GetMessages** can be used to read the current value. The value should be reset to the original value after use.*

Example

```
Saved = ax2ln.GetMessages;
ax2ln.Messages(FALSE);
ax2ln.SetFieldValue("Field","eee");
if (ax2ln.getNotesText()=='')
{...}
ax2ln.Messages(Saved);
```

Section

3.10.1

Other references

GetMessages

MoveFirst

Syntax

```
MoveFirst()
```

Input

None

Output

(Text)
Notes Document ID of the found document or error code.

Function

Finds and selects the <u>first</u> document in the last query result (***Query***). Returns the document's DocId or alternatively an error code.

Example

```
ax2ln.Query('Name="A*"');
ax2ln.MoveFirst();
while (StrLen(ax2ln.GetNotesText)>2)
{
    count++;
    PRINT "Count: ",ax2ln.QueryField('Count');
    ax2ln.MoveNext;
}
```

Error codes

"0":	No data in result-set
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.2

Other references

Query
MoveNext
MovePrev
MoveLast

MoveLast

Syntax

```
MoveLast()
```

Input

None

Output

(Text)
Notes Document ID of the found document or error code.

Function

Finds and selects the <u>last</u> document in the last query result (***Query***). Returns the document's DocId or alternatively an error code.

Example

```
ax2ln.Query('Name="A*"');
ax2ln.MoveLast();
while (StrLen(ax2ln.GetNotesText())>2)
{
    count++;
    print "Count: ",ax2ln.QueryField('Count');
    ax2ln.MovePrev();
}
```

Error codes

"0":	No data in result-set
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.2

Other references

Query
MoveNext
MovePrev
MoveFirst

MoveNext

Syntax

```
MoveNext()
```

Input

None

Output

(Text)
Notes Document ID of the found document or error code.

Function

Finds and selects the next document in the last query result (***Query***). Returns the document's DocId or alternatively an error code (e.g. if there are no more documents, "will be returned). Typically used after ***MoveFirst***.

Example

```
ax2ln.Query('Name="A*"');
ax2ln.MoveFirst();
while (StrLen(ax2ln.GetNotesText())>2)
{
    count++;
    print "Count: ",ax2ln.QueryField('Count');
    ax2ln.MoveNext();
}
```

Error codes

"":	No more data in result-set
"0":	No data in result-set
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.2

Other references

Query
MoveFirst
MovePrev
MoveLast

MovePrev

Syntax

```
MovePrev()
```

Input

None

Output

(Text)
Notes Document ID of the found document or error code.

Function

Finds and selects the previous document in the last query result (***Query***). Returns the document's DocId or alternatively an error code (e.g. if there are no more documents, "will be returned). Typically used after ***MoveLast***.

Example

```
ax2ln.Query('Name="A*"');
ax2ln.MoveLast();
while (StrLen(ax2ln.GetNotesText())>2)
{
    count++;
    print "Count: ",ax2ln.QueryField('Count');
    ax2ln.MovePrev();
}
```

Error codes

"":	No more data in result-set
"0":	No data in result-set
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.2

Other references

Query
MoveNext
MoveFirst
MoveLast

NextAttach

Syntax

```
NextAttach()
```

Input

None

Output

(Text: Direct)
Filename of the next attachment on the current document.

Function

Returns the filename of next attachment on the current selected document. This method is normally called after an initial call to ***FirstAttach*** and can be used to traverse through all attachments on a document.

If no more attachments are available an empty string is returned. Remark that error-codes can also be returned (1 or 2 characters).

Example

```
attach=ax2ln.FirstAttach();
while (StrLen(attach)>2)
{
    print 'Found attach: ',attach;
    ax2ln.DetachFile('c:\\temp\\'+attach,attach);
    attach=ax2ln.NextAttach();
}
```

Error codes

””:	No more attachments on document.
"2":	Form not selected, use method ***Form***
”3”:	Document not selected
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Other references

FirstAttach

NextField

Syntax

```
NextField()
```

Input

None

Output

(Text: Direct)
Name of the next field in the current document design.

Function

Selects the next field defined in the form (not the current document but the design) and returns the name of the field. Requires that ***FirstField*** has been called prior to use. Is typically used in a loop where all field names are read.

Example

```
f = ax2ln.FirstField();
while (f!='')
{
    print f;
    f = ax2ln.NextField();
}
```

Error codes

"":	No more fields on form
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

FirstField
NextForm
NextView

NextForm

Syntax

```
NextForm()
```

Input

None

Output

(Text: Direct)
Name of the next form in the current selected database.

Function

Selects the next form defined in the database and returns the name of the form. Requires that ***FirstForm*** has been called prior to use.
Names of both ordinary forms and subforms are returned.

Example

```
f = ax2ln.FirstForm();
while (f!='')
{
    print f;
    f = ax2ln.NextForm();
}
```

Error codes

"":	No more forms in database
"2":	No logon to database performed
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

FirstForm
NextField
NextView

NextView

Syntax

```
NextView()
```

Input

None

Output

(Text: Direct)
Name of the next view in the current selected database.

Function

Selects the next view in the current database and returns the name of this view. Requires that ***FirstView*** has been called prior to use.

Example

```
f = ax2ln.FirstView();
while (f!='')
{
    print f;
    f = ax2ln.NextView();
}
```

Error codes

"":	No more views in current database
"2":	No logon to database performed
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.9.2

Other references

FirstView
NextField
NextForm

OpenDialog

Syntax

```
OpenDialog([<Logon>])
```

Input

(Optional Boolean)
Optional parameter that specifies, that this command should not only browse but also logon to the server and database through the parameter TRUE.

Output

None

Function

This method can be used for browsing both between Notes servers and databases. Furthermore, if the additional parameter TRUE is specified, a session with the selected server and selected table will be established.

Example

```
ax2ln.OpenDialog(TRUE);     // Choose server
ax2ln.logon("Department"); // But not DB
```

Section

3.2.2.1 and 3.2.2.2

Other references

Server
Logon

OpenServer

Syntax

```
OpenServer(<Server name>)
```

Input

(Text)
Specification of server name.

Output

None

Function

Specification of the server to which to create link.
The server name "Local" specifies that the databases should be found on the local work-station.

Error codes

”1”:	Integration kit not initialized – call method ***Init***.
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.1

Other references

OpenDialog
ServerName

PutFieldValue

Syntax

```
PutFieldValue(<Fieldname>,<Fieldvalue>)
```

Input

(Text, Text)

1. Parameter:	Name of the Notes field whose value is to be set.
2. Parameter:	Value.

Output

None

Function

This method sets the field specified in the first parameter to the value specified in the second parameter. Unlike what is required with ***SetFieldValue***, the field does not have to be defined in the design of the current form.
The value (the second parameter) must always be of the type 'text'. Any other formats should be converted into 'text' before the method is called.
It is possible to specify that the field is to be saved with a different type in Notes (since the field is not necessarily defined in the form). This is done by prefixing the field name by #, data type, followed by #(see example and Appendix 4).
Date-values <u>must always</u> have the format "dd-mm-yy" or "dd-mm-yyyy" – using dash as delimiter.

Example

```
ax2ln.SelectID(F.NotesDocID);
S = Date2Str(F.Dato,123,2,3,2,3,2);
ax2ln.PutFieldValue("#DATETIME#Date",S);
ax2ln.Commit();
```

Error codes

"3":	Syntax error.
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.5.2

Other references

SetFieldValue

Query

Syntax

```
Query(<Notes SELECT command>)
```

Input

(Text)
A Notes SELECT command, to be used for selecting Notes documents. If a blank (“”) command is specified, all documents in the current selected view is selected. The word SELECT is optional.

Output

(Comma-separated file and in-memory result set)
The result of the query is returned in a comma-separated file and as a result set in memory.

Function

This method is used for general queries for Notes documents. If no parameters are specified, all documents are returned in the current view. This may e.g. be used when views are available in Notes, which make the necessary selection.
Any parameter specified must be a valid Notes selection. Notes formulas can be used (e.g. "SELECT @All" or just "@All").
If a temporary file has been assigned, data is also stored in this file as comma-separated values. Remember to delete the comma-delimited file after use.
Data is also available as an in-memory result-set.
Use ***MoveFirst***, ***MoveNext***, ***MovePrev*** and ***MoveLast*** to navigate through this result set or read the comma file.

Example

```
ax2ln.Query(‘@contains(Number;"1")’);
```

Error codes

"0":	No data found.
"1":	Error creating or writing to comma separated file
"9":	Form or view not selected, use ***Form*** or ***View***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.1

QueryField

Syntax

```
QueryField(<Fieldname>)
```

Input

(Text)
Specification of which field in Notes to receive the value from.

Output

(Text: Direct)
Field value

Function

Query on the current document for field value. The document must have been selected prior to query. Remember that specification of field name must be exact (Notes is case-sensitive). *Return values are always in text*. If another value type is retrieved, e.g. date or numerical value, this must subsequently be converted.

Example

```
ax2ln.GetUnique("Number","123");
print "Dep. name: ", ax2ln.QueryField("Name");
print "Notes Ref: ", ax2ln.QueryField("Ref");
```

Error codes

"1":	Field is not defined on Notes form
"2":	Form not selected, use method ***Form***
"3":	No document selected
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.5

Other references

SetFieldValue

QueryRichTextField

Syntax

```
QueryRichTextField(<Fieldname>)
```

Input

(Text)
Specification of the name of the Notes rich-text field whose value should be fetched.

Output

(Text: Direct)
The field value.

Function

Query in the current document for a field value. The document must be selected before the query. Remember that the field name specification must be exact (Notes is case sensitive).
The return value is always in text format.
Unlike what happens when ***QueryField*** is used the value will also be returned to the temporary file (see ***SetTmpFileName***). This way, it is possible to read the contents of rich text fields spanning multiple lines.

Example

```
ax2ln.GetUnique("Number","123");
ax2ln.GetUnique("Number","123");
Print ax2ln.QueryRichTextField('RT');
```

Error codes

"1":	Field is not defined on Notes form
"2":	Form not selected, use method ***Form***
"3":	No document selected
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.5

Other references

QueryField

QueryUnique

Syntax

```
QueryUnique(<Notes SELECT kommando>)
```

Input

(Text)
Specification of Notes SELECT command to be used for selection of <u>one</u> Notes document.

Output

(Text)
Document ID of the found Notes document or error code

Function

This method is used for query for one specific Notes document, which can be selected via a valid SELECT statement.
Any parameter specified must be a valid Notes selection. It is regarded an error if more than one document is returned. The method is very useful for queries via more AX key fields (see example below).

Extended functionality:
By prefixing the first parameter with 'ANY@@@' the command will not fail if more documents fulfilling the search-criteria is found. A pointer to the first document is returned.
By prefixing the first parameter with 'MANUAL@@@' the query will not be fixed to documents of the current form. Any document complying with the search-criteria will be returned, regardless of the form used on the document.

Example

```
s = 'Proj+Proj+'" & Section="'+Section+'"';
ax2ln.QueryUnique(s);
if (StrLen(ax2ln.GetNotesText())>2)
{
...
```

Error codes

"0":	No data found.
"1":	Form not selected, use method ***Form***
"3":	More than 1 document found
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.3

Other references

Query

QueryView

Syntax

```
QueryView()
```

Input

None

Output

(Comma-separated file and in-memory result set)
The result of the query is returned in a comma-separated file and in a result set in memory.

Function

This method corresponds to the ***Query*** method, but with this method no SELECT string is used. Instead all documents in current view are returned in a result-set and in a comma-separated file (if a temporary file has been assigned).

Example

```
ax2ln.view('NewDocs');
ax2ln.QueryView();
```

Error codes

"0":	No data found.
”1”:	Error creating or writing to comma separated file
"9":	View not selected, use method ***View***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.1

Other references

View
Query

RemoveItem

Syntax

`RemoveItem(<ItemName>)`

Input

(Text)
The name of the item to be removed from current document

Output

None

Function

This method is used to remove an item on the current document. The item does not have to exist as a field on the current form.
If the item is a multi-value item all values will be deleted.

Example

```
Ax2ln.RemoveItem('Number');
```

Error codes

"2":	Document not selected
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Other references

Query

SearchFirst

Syntax

```
SearchFirst()
```

Input

None

Output

(Text)
The Notes Document ID of the found document or an error code.

Function

Returns the first document of the current view. Using Query by View makes searching much faster than searching by Query. The view must be open.

The View must be designed in such a way that the first column is sorted and the first column MUST be of the type 'text' and must not contain columns defined as a "constant".

Example

```
ax2ln.SearchFirst();
while (StrLen(ax2ln.getNotesText())>2)
{
    Print "Name: ", ax2ln.QueryField('Name');
    ax2ln.SearchNext();
}
```

Error codes

"0":	No data found.
"1":	View not selected, use method ***View***
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.4

Other references

SearchNext
SearchView

SearchNext

Syntax

```
SearchNext()
```

Input

None

Output

(Text)
The Notes Document ID of the found document or an error code.

Function

Returns the next document in the current view. Using Query by view makes searching much faster than searching by query.
To be used typically after ***SearchFirst*** or ***SearchView***.
The view must be open.
Remark that if this method is used after a call to ***SearchView*** only documents matching the ***SearchView*** can be traversed.

Example

```
ax2ln.SearchFirst();
while (StrLen(ax2ln.getNotesText())>2)
{
    Print "Name: ", ax2ln.QueryField('Name');
    ax2ln.SearchNext();
}
```

Error codes

"0":	No more documents
"1":	View not selected, use method ***View***
"2":	Form not selected, use method ***Form***
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.4

Other references

SearchFirst
SearchView

SearchView

Syntax

```
SearchView(<Search value>)
```

Input
(Text)
Specification of the value to be searched for in the view.

Output
(Text)
The Notes Document ID of the found document or an error code.

Function
Returns the first document in the current view which matches the search value.
Using ***SearchView*** makes searching much faster than searching by ***Query*** (full-text).
Should always be used in connection with lookups in views with large amounts of documents.
The view must be open.

The view must be designed in such a way that the first column is sorted and the first column MUST be of the type `'text'` and must not contain columns defined as a “constant”.

If there is more than one search key, the search keys must be separated by a separator (default is semi-colon (”;”)). Furthermore, columns in views in Notes must match the parameter order.

Example

```
ax2ln.SearchView('Doe;John');
while (StrLen(ax2ln.GetNotesText())>2)
{
    Print "Name: ",ax2ln.QueryField('Name');
    ax2ln.SearchNext();
}
```

Error codes

"0":	No data found.
"1":	View not selected, use method ***View***
"9":	Form not selected, use method ***Form***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.4

Other references

SearchNext
SearchView
SetDelim

SelectID

Syntax

```
SelectID(<DocID>)
```

Input

(Text)
Notes Document ID on document to be selected.

Output

None

Function

Direct selection of Notes document via known DocId. The DocId may e.g. have been found via ***Query*** in which DocId is contained as the first field in each line of the comma-delimited file.

This method is very useful in cases where a reference to a specific Notes document has been stored in an AX table field.

Example

```
ID = Afdeling.NotesDocID;
ax2ln.SelectID(ID);
```

Error codes

"1":	Wrong DocId (e.g. invalid ID)
"2":	Form not selected, use method ***Form***
"3":	Document not found
”4”:	Form name missing on found document
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.4.6

Other references

DeleteID
Query

SelectUNID

Syntax

```
SelectUNID(<UNID>)
```

Input

(Text)
The Notes Universal Document ID of the document to be found.

Output

None

Function

Direct query for Notes document using known 32 bytes Universal Document ID. This UNID can be found by using e.g. ***UNID***. The UNID should be used in stead of DocID when working across replicas.
Remember to reserve 32 characters for UNID's in AX's string variable.

Error codes

"2":	Form not selected, use method ***Form***
"3":	Document with specified UNID not found
"99":	"Exception error": Other error from Integration Kit. See Error Message in Window.

Section

3.4.6

Other references

UNID
SelectID

ServerName

Syntax

```
ServerName()
```

Input

None

Output

(Text: Direct)
Name of current selected server

Function

Returns the name of the current server selected.

Example

```
print "Current server: ", ax2ln.ServerName();
pause;
```

Section

3.3.1

Other references

OpenServer

SetComputeWithForm

Syntax

```
SetComputeWithForm(<compute>)
```

Input

1. Parameter: Indication whether formulas should be executed or not

FALSE:	Do NOT execute formulas when calling ***Commit***
TRUE:	Calculate formulas when calling ***Commit***.

Output

None

Function

When calling the method ***Commit*** by default no formulas on form are executed. By calling ***SetComputeWithForm*** this behaviour can be changed, so formulas are executed – as by calling ***CommitWithForm.***

Example

```
ax2ln.init();
ax2ln.SetComputeWithForm(TRUE);
ax2ln.CreateNew();
ax2ln.PutFieldValue('TextItem', 'Hello World!');
ax2ln.Commit();
```

Other references

Commit
CommitWithForm
CreateNew

SetDelay

Syntax

```
SetDelay(<Delay in ms>)
```

Input

(Number)
Specification of delay in milliseconds between each DLL Notes call.

Output

None

Function

To avoid overloading Lotus Domino servers in connection with updates, a break between each DLL call to Lotus Notes/Domino can be inserted by ***SetDelay***. The break must be specified in milliseconds – typically 10-100 ms. This way, the update will be a little slower, but the server's becoming very slow will be avoided.
Setting the parameter to "0" de-activates the delay.

Example

```
ax2ln.SetDelay(100);
ax2ln.SearchFirst();
while (StrLen(ax2ln.GetNotesText()) > 2)
{
    ax2ln.SetFieldValue("Name",DebTable.Name);
    ax2ln.Commit();
    ax2ln.SearchNext();
}
ax2ln.SetDelay(0);
```

SetDelim

Syntax

```
SetDelim(<Delimiter character>)
```

Input

(Text)
Character to be used as search-field separator.

Output

None

Function

Semi-colon is the default separator in connection with the transfer of multiple search values to ***SearchView***. If this cannot be used, an alternative separator can be set by using ***SetDelim***.

Section

3.4.4

Example

```
QStr=CustTable.Name+'@'+CustTable.ZipCode;
ax2ln.SetDelim('@');
ax2ln.SearchView(QStr);
```

SetFieldValue

Syntax

```
SetFieldValue(<Fieldname>,<Fieldvalue>[,<COMMIT>])
```

Input

(Text, Text[, Boolean])

1. Parameter:	Field in Notes to be updated
2. Parameter:	Value to update with
3. Parameter:	Optional selection of commit

Output

None

Function

This method sets the field specified in the first parameter to the value specified in the second parameter. Value (second parameter) must always be of the type text. Any other formats must be converted to text before the method is called. Existing field contents are overwritten by this method.

Date-values must always have the format “dd-mm-yy” or “dd-mm-yyyy” – using dash as delimiter.

If TRUE is specified as the third parameter, the value is stored in Notes immediately. This is useful in the case of a single update. If more fields are subsequently to be updated, ***Commit*** should be called separately as a final command.

Example

```
ax2ln.SelectID(F.NotesDocID);
ax2ln.SetFieldValue('Date',Date2Str(F.Dato,123,2,3,2,3,2));
ax2ln.commit();
```

Error codes

"1":	Field not found in form
"2":	Form not selected, use method ***Form***
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.5.2

Other references

AppendFieldValue
SetRichFieldValue
QueryField

SetHandle

Syntax

```
SetHandle([<Handle number>])
```

Input

(Number)
Optional specification of handle for Lotus Notes

Output

None

Function

In case of need to use multiple open sessions in Lotus Notes, this can be achieved by using multiple handles. The integration kit supports up to 15 current handles.
Use SetHandle to switch between handles. Each session has its own connection with Lotus Notes.

Example

```
ax2ln.SetHandle(2);
ax2ln.OpenServer("Local");
ax2ln.Logon("MyDB.nsf");
ax2ln.Form("Form 2");
ax2ln.View("View 2");
//Return to previous session
ax2ln.SetHandle(1);
```

Section

3.2.4

Other references

GetHandle

SetKeepUnread

Syntax

```
SetKeepUnread(<Keep>)
```

Input

(Boolean)
Specification of whether a check is required to see if field should be overwritten.

Output

None

Function

Normally, the integration kit will overwrite existing data regardless of any differences between the new field value and the old one. This means that the document will be marked as unread and LAST_MODIFIED will be changed. In certain situations, however, data may only be overwritten if the field values do differ. ***SetKeepUnread*** handles this. If it is activated the field value will be read before each write and writing in Notes will only take place if the new field value is different from the old one.

Example

```
Ax2ln.SetKeepUnread(TRUE);
```

Section

3.4.7

Other references

CompareField

SetListDelim

Syntax

```
SetListDelim(<Delimiter Character>)
```

Input

(Text)
Character to be used to delimit values in a multi-value field.

Output

None

Function

AX does not, as Lotus Notes, support multi-value fields. To transfer values (e.g. by ***SetFieldValue***) to a multi-value field each value can be delimited by a delimiter character. The default delimiter in the integration kit is ";". If this character is not suitable (e.g. when transferring values including semi-colon in the value), this delimiter character can be changed by using ***SetListDelim***.

Example

```
S = '';
ax2ln.SetListDelim('@');
while select EmplTable
where (EmplTable.CountryId == '')
    s = s+EmplTable.Name+'@';
ax2ln.SetFieldValue('LocalContacts',s);
```

Other references

SetFieldValue
AppendFieldValue
AppendTextList

SetLog

Syntax

```
SetLog(<Log>)
```

Input

(Boolean)
Specification of whether the logging of debug info is to take place in a file.

Output

None

Function

Sometimes it is a good idea to be able to trace the debug information without having to monitor the computer. ***SetLog*** activates logging to a file in stead of to the screen. Remember to specify the name of the log file and to de-activate file logging when the de-bugging is over since all messages will be written in a file and thus not shown on the screen.

Example

```
ax2ln.SetLogFileName('debug.log');
ax2ln.ClearLog();
ax2ln.SetLog(TRUE);
ax2ln.Debug(TRUE);
ax2ln.SearchView('1234');
ax2ln.SetLog(FALSE);
```

Section

3.10.3

Other references

ClearLog
SetLogFileName

SetLogFilename

Syntax

```
SetLogFileName[(<Log filename>)]
```

Input

(Text)
Specification of the file in which debug logging must take place.

Output

None

Function

In connection with the use of ***SetLog***, this method is used to specify the name of the file in which the log information must be written.
If an empty string is supplied, the default log file name is used (can be defined in erp2ln.ini).

Example

```
ax2ln.SetLogFileName('debug.log');
ax2ln.ClearLog();
ax2ln.SetLog(TRUE);
ax2ln.Debug(TRUE);
ax2ln.SearchView('1234');
ax2ln.SetLog(FALSE);
```

Section

3.10.3

Other references

SetLog
ClearLog
Debug

SetRichSepMode

Syntax

```
SetRichSepMode(<SepMode>)
```

Input

(Boolean)

Specification of how the Integration Kit handles rich-text fields when using ***Query***.

FALSE:	Fields in primary comma separated result file
TRUE:	Fields in separate text-files.

Output

None

Function

If parameter is FALSE, the first line of each rich-text field is stored in the primary comma-delimited file in connection with ***Query***.
If parameter is TRUE, rich-text fields will not be placed in the comma-delimited file returned as primary outcome on the call of ***Query***. Instead it will be placed in separate comma-delimited files in the subdirectory "RichText". One file per rich-text field per document.
See section 3.6.2.2 for a more detailed description.

Section

3.6.2.2

Other references

GetRichSepMode

SetTmpFileName

Syntax

```
SetTmpFileName([<Filename>])
```

Input

(Text)
Optional specification of name on temporary file.

Output

The name of the temporary file is returned with ***GetNotesText***

Function

This method is used before call of Query for specification of the file name of the temporary comma-delimited file in which the query result is returned. If a parameter is specified, this will be used as file name. Normally no parameter is specified in which case the integration kit "computes" a unique file name. After call of ***SetTmpFileName***, the file name can be read With the method ***GetNotesText***.
Remember to delete the temporary file after use.

Example

```
str 60 Filename = 'c:\\temp\\test.$$$';
...
ax2ln.SetTempFileName(FileName);
ax2ln.Query('select @All')
...
ax2ln.DeleteTmpFile();
```

Section

3.3.3

Other references

Query

ShowAll

Syntax

```
ShowAll()
```

Input

None

Output

None

Function

Opens a window with miscellaneous information about the integration kit and the current session:

- Version number
- Any restrictions (e.g. only mail.-functionality)
- Miscellaneous information regarding serial number
- Session number
- File name of temporary file
- Current Notes.ini file used

Example

```
ax2ln.Init();
ax2ln.ShowAll();
```

Section

3.3.1

UIOpenDocument

Syntax

```
UIOpenDocument()
```

Input

None

Output

None

Function

Opens the currently selcted document in the Lotus Notes client.

Example

```
ax2ln.OpenServer("Local");
ax2ln.Logon("MyDB.nsf");
ax2ln.Form("Form 2");
ax2ln.View("View 2");
ax2ln.SearchFirst();
ax2ln.UIOpenDocument();
```

Other references

MailComposeOLE

UNID

Syntax

```
UNID()
```

Input

None

Output

(Text: Direct)
Notes Universal Document Id of active document in Notes.

Function

Returns the Notes Universal Document Id of the active document in Notes. The Notes Universal Document Id is a 32-character string which is unique in a database across replicas.

Example

```
ax2ln.selectId('00001A22');
print ax2ln.UNID();
pause;
```

Section

3.5.5

Other references

SelectUNID
DocID

View

Syntax

```
View(<View name>)
```

Input

(Text)
Name of view in Lotus Notes to use.

Output

None

Function

Specification of the view in Notes that is to be used for queries with ***Query*** without parameters.
If view name is contained in parenthesizes (hidden view), these parenthesizes shall not be used in view name.

Example

```
ax2ln.View('All');
print "Current view: ",ax2ln.ViewName();
ax2ln.SearchView(MyKey);
```

Error codes

"1":	View not found in database
"9":	Logon to database is missing
"99":	“Exception error”: Other error from Integration Kit. See Error Message in Window.

Section

3.2.2.3

Other references

ViewName

ViewName

Syntax

```
ViewName()
```

Input

None

Output

(Text: Direct)
Name of current view

Function

Returns the selected current view in Lotus Notes, defined via ***View***.

Example

```
print "Current view: ",ax2ln.ViewName();
pause;
```

Section

3.3.1

Other references

View

Appendix 1. Troubleshooting

Checklist

1. Check if the path to the binary files in Lotus Notes and Notes.ini are located in the system variable PATH.
2. Check that there is an individual Notes.ini file for all users. The Integration Kit reads information about the user's mail settings. This requires a Notes.ini for each user in e.g. a Citrix-environment.
3. If you have a Lotus Notes client version 6 or you recently up-graded your Lotus Notes client 5 to a 6 the DLL 'nlsxbe.dll' might have to be reregistered. Do this by opening the Windows start-menu, select Run and enter the following (without the single quotes): 'regsvr32 c:\lotus\Notes\nlsxbe.dll'. Remember to replace the path 'c:\lotus\Notes' with the path to you installation of the Lotus Notes client.
4. If both Lotus Notes 5 and 6 installed, this can cause problems. The Integration Kit uses a COM-object which is registered by Lotus Notes on installation. If Notes 6 is installed afterwards it will overwrite part of Notes 5's registration. Try to reinstall Lotus Notes 5 to see if the problem disap-pears. As a rule of thumb, the Notes client installed last must be the one to be used for the Inte-gration Kit.
5. If Dynamics AX freezes during startup (after the user name and pass-word have been entered), the kernel version of AX must be checked. The kernel (ax32.exe) must be version 2.5.1270.3703 or later. Earlier versions cause problems if the Windows operating system is updated with the lat-est security updates. This is a general error in AX.
6. Check that the correct AX client is being used on startup. The Integrations Kit depends on it being the correct client since Integration Kit-files are located in the bin-directory of the client.
7. Check if there is more than one Notes.ini. The Notes.ini which is active and is being used by the Integration Kit can be found by selecting the Windows Start menu and clicking run. Enter Notes.ini and click [OK]. You can also search the lo-cal workstation for Notes.ini-files. The Ini-file may be located in a network drive. If the active Notes.ini is the wrong one, errors may occur in AX when a synchronization job is being run. The error is that it will constantly say that the user in AX is not identical with the user in Lotus Notes (the username is found in Notes.ini).
8. If you use the distribution program 'Snow', you must be aware that the internal function 'Regis-terDLL' does not work properly in all Window versions. Use Windows' own version 'regsvr32.exe' in stead (use the parameter /s for run 'silently').
9. Windows will err when you register smmDrop2.ocx and smmWrap.ocx in Windows 9x. It has been reported that the ocx-files mentioned must be registered from the Dynamics AX client's bin-directory. Therefore, you must copy the file regsvr32.exe to the bin-directory before executing it. This is not an error in the Integration Kit.

Appendix 2. Known errors and limitations

2.1 Cleanup at errors

Must of the possible errors that can occur when integrating Dynamics AX with Lotus Notes, is handled by the Integration kit. Error codes can normally be retrieved by the method ***GetNotesText***.
But from time to time there can be special case, where errors are not “caught” by the Integration kit. These errors occur outside from AX, in the connection between the DLL and the Lotus Notes API. If possible the integration kit will display a Windows error message and return with error code “99”. There may be some situations where the Integration kit can not handle the error. It may be necessary to shutdown the AX windows-process (Ctrl-Alt-Delete or Windows Task manager).

If the Integration kit is shutdown this way no memory cleanup is performed. In some cases this may lead to instability when performing AX integration with Lotus Notes afterwards.
Restart the client if this occurs.

2.2 Multi-value fields at Query

Lotus Notes supports multi-value fields. This functionality is not supported in the same way in AX. When requesting such a value from Lotus Notes, the values are returned delimited by semi-colon (or another delimiter set by SetListDelim).

Caution should be taken to handle return values from multi-value fields. The AX code should handle the split of values to e.g. separate records in AX.

The method ***AppendTextList*** does only support text multi-value fields.

2.3 Mail-error at errors in mail-integration

Errors in the mail-methods can cause problems with the functionality regarding the method MailSetError.

2.4 Attachments in rich-text fields

When deleting an attachment on a rich-text field, the icon is not removed. The attachment is removed.

2.5 Limitations in graphics formats

The Integration kit does not support all graphic formats in the methods MailBodyFile and ImportFile.
The following formats are supported:

- ASCII, standard ASCII-files (TXT, PRN,ASC...)
- Rich-text files (RTF).
- PCX-files
- BMP-files
- TIFF-files

These formats can vary in their formats – not all variants are necessarily supported. It is recommended, that the compatibility is tested for the necessary formats.

2.6 DocId at replicas

It is important to know, that DocId's is only unique inside one database. If the database is replicated to another server, the DocId could change. If DocId is used as an identifier in the AX code, it is important that the same database is used always. It is better to use GetUnique, QueryUnique, SearchView or other methods, that uses data-keys for lookup.
Another option is to use UNID as a key. This value is unchanged if data is replicated to other databases and servers.

2.7 Length of field values

The integration kit does not support field values larger than 32.000 characters (RichText fields can be larger, by adding more content in sequence).

2.8 The field "Form" must be present on documents

The integration kit does, in most cases, rely on information from the form-design. Therefore the field form must always be present in documents handled by the Integration kit.

Appendix 3. Type of design elements

With the methods FieldType and FormType it is possible to get information regarding fields and forms from the design in Lotus Notes.

The following field types can be returned from FieldType:

- Unknown
- Error
- Text
- Number
- DateTime
- RichText
- TextList
- NumberList
- DateTimeList

The following form types can be returned from FormType:

- Main
- Response to Main
- Response to Response

Appendix 4. Data type parameters

The following methods, who operates on ”items”, can be called with a special prefix in first parameter that indicates which data type the item should be stored as in Lotus Notes:

- ***PutFieldValue***
- ***AppendFieldValue***
- ***GetFieldLength***
- ***GetFieldValue***

The following types can be used:

TEXT	use #TEXT#
NUMBER	use #NUMBER#
DATETIME	use #DATETIME#
TEXT_LIST	use #TEXT_LIST#
NUMBER_LIST	use #NUMBER_LIST#
DATETIME_LIST	use #DATETIME_LIST#

Examples:
PutFieldValue(’#NUMBER#Amount”,”123,45”)
AppendFieldValue(’#DATETIME_LIST#Dates’,TodayStr)

Remember to use the format “dd-mm-yy” or “dd-mm-yyyy” on date items.

Appendix 5. INTOGRATE in a Terminal/Citrix environment

To install INTOGRATE in a Terminal/Citrix environment the following software must be installed on the server:

- Lotus Notes client (R5 or later)
- Dynamics AX client

A common setup for Dynamics AX would be to install a local AX client on the Terminal/Citrix-server (to optimize for speed).

The usual way of deploying Lotus Notes client on Terminal/Citrix is described in the following Redpaper from IBM (see chapter 5.4.3 or 5.4.4 depending on your version of Lotus Notes):
http://www.redbooks.ibm.com/abstracts/redp3629.html
More specific details about installing Lotus Notes on a Terminal/Citrix server should be obtained from your Lotus Notes vendor.
In this scenario the Lotus Notes client is installed in "C:\Program Files\Lotus\Notes" and each user's data-directory is "H:\Notes\Data" (containing both the data files and the notes.ini file).

When Dynamics AX and Lotus Notes are ready for use, INTOGRATE can be installed.
The procedure is as follows:

1. Install INTOGRATE either by running the installation program or by using the manual procedure described in section 2.
2. Configure the system environment PATH to include "H:\Notes\Data".
3. Install INTOGRATE in AX by importing the XPO-files.

When running INTOGRATE on Terminal/Citrix, make sure the following settings are enabled in erp2ln.ini (the same directory that contains the ERP2LN.DLL file):

```
DisableNotesSetupDialog=1
DisableRegistryCheck=1
DisableRegistryAutoFix=1
```

These settings will prohibit INTOGRATE from modifying the Windows registry on the server.
A few hints:

- Make sure there is only one notes.ini for each user (sometimes there is a notes.ini elsewhere in the system/user environment path)
- Make sure you have the latest version of INTOGRATE (at the time of writing the current version is 1.6.0.343).

Try to add the Lotus Notes binary directory to the system PATH (e.g. "C:\Program Files\Lotus\Notes"). This might solve some issues with the Lotus Notes setup.

Appendix 6. Adjustments to Microsoft Dynamics AX

The below table describes all the elements in Microsoft Dynamics AX which are modified or added as part of "INTOGRATE AX".

All modifications made to existing elements in Microsoft Dynamics AX are commented in the following manner:

// INTOGRATE AX: START
.... code modifications
// INTOGRATE AX: END

Modifications to local and global variables are not described in the code, but can be seen in the text versions of the import files. These can be found on the INTOGRATE CD or in the installation directory.

New elements

Object type	Object – method	Project
Class	AX2LN	I2I_LN
Class	AX2LN_addon	I2I_LN

Appendix 7. Manual installation of the Integration Kit

It is possible to install the Integration Kit without using the installation program. This may be advantageous if the installation is to be carried out in a large environment and the files are to be distributed to multiple workstations or an advanced installation is involved e.g. a Citrix-based network.

Unpacking files

If the installation kit is packed into one file, it must first be unpacked. This is done by starting “INTOGRATE AX - CRM x.x.x.x.exe” and selecting the drive in which the files are to be unpacked. The destination of the files must be a local drive or a network drive.

Kommentar [AHH1]: Mangler her ikke noget?

Copying files

When the files have been unpacked they must be distributed to the relevant directories.
The following files must be copied to ”…\Microsoft Dynamics AX\40\Client\Bin” (they can be found in ”data\files” in the installation directory).

ERP2LN.DLL
ERP2LN_ADDON.DLL
ERP2LN.INI (optional)
I2ICOMP.OCX

Setup of PATH

For the integration to work properly Lotus Notes’ binary files and notes.ini must be located in the system variable ”PATH”. In our example, both the binary files and notes.ini are located in ”C:\LOTUS\NOTES”. If this is different in your installation, use your settings instead.
Setting up the PATH in newer versions of the integration kit is most likely not necessary. The location of the Lotus Notes binary-directory and notes.ini can usually be found in the Windows registry. Based on the registry the integration kit will automatically retrieve the correct locations. Note: in environments where users cannot edit the registry on the local computer you must add the Lotus Notes binary directory to the PATH as described above.
If you plan to start Lotus Notes with a direct path to a notes.ini (ex. “notes.exe =m:\notes\notes.ini”) the registry must also be set up accordingly.
When the integration kit starts up, the location of the binary directory and notes.ini are read from the following key in the registry:

```
HKEY_CLASSES_ROOT\CLSID\{29131502-2EED-1069-BF5D-00DD011186B7}\LocalServer32
```

An example value could be:

```
(Default)="C:\Program Files\lotus\notes\notes.exe =m:\notes\notes.ini" /Automation
```

The reason why notes.ini must be set in this registry is that Lotus Notes can be started from several locations that do not use the shortcut with a direct path to notes.ini. An example of this is when a user clicks on a mailto-link on an internet page. The browser will use the registry to start Lotus Notes, and not a shortcut on the desktop or in the programs menu.

Check PATH configuration

To check if PATH has been correctly set up, you must start an MS-DOS prompt. In MS-DOS, type ”Path” and press return. The operating system will now present a list of the directories in the current setup separated by semi-colons. Check that both the directory for the binary files and the one for notes.ini are included in the list separated by semi-colons.

Setup of PATH in Windows 9x

If the path to the binary files or notes.ini is missing, Autoexec.bat must be modified in the following way:

1. Start the text editor in an MS-DOS prompt by typing ”EDIT C:\AUTOEXEC.BAT”.
2. Find the line that begins with ”PATH=”
3. If the line exists, add the following: ”;C:\LOTUS\NOTES”. Example of correct PATH:

   ```
   PATH=C:\WINDOWS;C:\WINDOWS\COMMAND; C:\LOTUS\NOTES;
   ```

4. If the PATH line does not exist, then add the following at the end of AUTOEXEC.BAT:

   ```
   PATH=%PATH%;C:\LOTUS\NOTES;
   ```

5. Save the changes by selecting ’Save’ in the ’File’ menu.
6. Close the text editor by selecting the menu ’File’ and the item ’Exit’.
7. Exit the MS-DOS command prompt by typing ”EXIT” and pressing return.
8. Restart you computer and check the ”PATH” settings by means of the procedure described in the section ’Check PATH configuration’.

Setup of PATH in Windows NT/2000

1. Right-click ’My computer’ on your desktop and select ’Properties’.
2. Select the ’Advanced’ tab and click the button ’Environment Variables…’.
3. Under ’System Variables’, select the line that includes the word ”Path”.
4. Click ’Edit’ and check if the path to ”C:\LOTUS\NOTES” is there. If not then add it at the end of the line (remember to insert a semi-colon before you add the directory name to the path).
5. Save the settings by clicking the [OK] button in the three following dialog boxes.

Setup of the Windows registration database

The component 2ICOMP.OCX must be registered in the Windows registration database. This is done as follows:

1. Click the Windows start button and select the menu item ’Run…’.
2. Enter the following in the textbox and click the [OK] button:

   ```
   regsvr32 “c:\Navision\Microsoft Dynamics AX Client\bin\I2ICOMP.OCX”
   ```

3. A box with the text "DLLRegisterServer in x:\...... succeeded" will appear. This indicates that the component is correctly installed.

Kommentar [AHH2]: Ser ud som om I selv har rettet dette til 6,7,8. Så derfor gør jeg intet. Men skulle det ikke være 1,2,3?

It is possible to run this part of the user installation without user interaction. Use the parameter ”/s” on regsvr32.exe. Example:

```
regsvr32 “c:\Navision\Microsoft Dynamics AX\40\Client\Bin \I2ICOMP.OCX” /s
```

Appendix 8. Troubleshooting

Checklist

1. Check if you can send an email from your Lotus Notes client. Try starting your Lotus Notes client and pressing <CTRL>+<M> to create a new memo. If this does not work, check the settings in your location document (check that a mail server and a mail file have been specified).
2. Check if the path to the binary files in Lotus Notes and notes.ini are located in the system variable PATH. See Appendix 7 for help with this.
3. Check that there is an individual notes.ini file for all users. The Integration Kit reads information about the user's mail settings. This requires a notes.ini for each user in e.g. a Citrix-environment.
4. If you have a Lotus Notes client version 6 or you recently upgraded your Lotus Notes client 5 to a 6 the dll 'nlsxbe.dll' might have to be reregistered. Do this by opening the Windows start-menu, select Run and enter the following (without the single quotes): 'regsvr32 c:\lotus\notes\nlsxbe.dll'. Remember to replace the path 'c:\lotus\notes' with the path to your Lotus Notes client.
5. If both Lotus Notes 5 and 6 (or 7) are installed, this can cause problems. The Integration Kit uses a COM-object which is registered by Lotus Notes on installation. If Notes 6 is installed afterwards it will overwrite part of Notes 5's registration. Try to reinstall Lotus Notes 5 to see if the problem disappears. As a rule of thumb, the Notes client installed last must be the one to be used for the Integration Kit.
6. Check that the correct Microsoft Dynamics AX client is being used on start-up. The Integration Kit depends on it being the correct client since Integration Kit-files are located in the bin-directory of the client.
7. Check if there is more than one notes.ini. The notes.ini which is active and is being used by the Integration Kit can be found by selecting the Windows Start menu and clicking run. Enter notes.ini and click [OK] (if this does not find a notes.ini file, then see the section on the setup of PATH on page 174). You can also search the local workstation for notes.ini-files. The Ini file may be located in a network drive. If the active notes.ini is the wrong one, errors may occur in Microsoft Dynamics AX when a synchronisation job is being run. The error symptom is a constantly reappearing message saying that the user in Microsoft Dynamics AX is not identical with the user in Lotus Notes (the username is found in notes.ini).
8. If Microsoft Dynamics AX says that "ERP2LN.DLL" cannot be found, make sure that the 'alt. bin directory' in the Microsoft Dynamics AX configuration is set up to point to the Microsoft Dynamics AX client's bin-directory.
9. If you use the distribution program 'Snow', you must be aware that the internal function 'RegisterDLL' does not work properly in all Windows versions. Use Windows' own version 'regsvr32.exe' in stead (use the parameter /s for running it 'silently').
10. Windows will cause an error when you register smmDrop2.ocx and smmWrap.ocx in Windows 9x. It has been reported that the ocx-files mentioned must be registered from the Microsoft Dynamics AX client's bin-directory. Therefore, you must copy the file regsvr32.exe to the bin-directory before executing it. This is not an error in the Integration Kit.

Kommentar [AHH3]: Samme som før. I har rettet til start bullet 9, men hvorfor skal den ikke starte med 1?

Appendix 9. Configuration - ERP2LN.INI

Logging system

INTOGRATE has the option to write detailed error messages, warnings and information to a log file, helping to identify most issues.
This is configured using INTOGRATE’s configuration file “ERP2LN.INI”.
“ERP2LN.INI” is located in the same directory as “ERP2LN.DLL”. The exact location depends on the installed INTOGRATE product (typically “C:\Program Files\INTOGRATE Navision”).
It is possible to select the level of detail to write to the log file. This is controlled by the “FileLogLevel” setting. Possible values are as follows:

0	Nothing is logged
1	Errors
2	Information
4	Debug
8	Trace/Debug (will cause a severe reduction of performance)

To enable several levels simply add the numbers together. For instance to enable “Errors” and “Information”, FileLogLevel must be set to 3 (1 + 2). To enable full logging use the value 15.
Procedure to change “ERP2LN.INI” (lines beginning with semicolon are considered comments and will not be processed by INTOGRATE):
Open “ERP2LN.INI” in Windows Notepad (or similar text editor).
Search for the word FileLogLevel (this will locate the section where the file log level is set, showing examples of use etc.).
Add a new line

```
FileLogLevel=XX
```

where XX is the level of details to log (use 15 to enable full logging).
Save and close the file.
Restart Navision to activate the changes.
The log file has a default location of “C:\Documents and Settings\<username>\Local Settings\Temp\ERP2LN” where “<username>” is the currently logged on Windows user.
This can be changed in “ERP2LN.INI” by setting the LogDirectory to an alternative location (using the same procedure as with FileLogLevel).