Revised 29.March 2001

LiSA-Interface

A component of the Lotus i-net Solution Architecure



Reference Manual



Table of Contents

Introduction

- **Interface Archetypes**
- **Textual serialization of a business object's data**
- **View design philosophy**

LiSA-interfaces

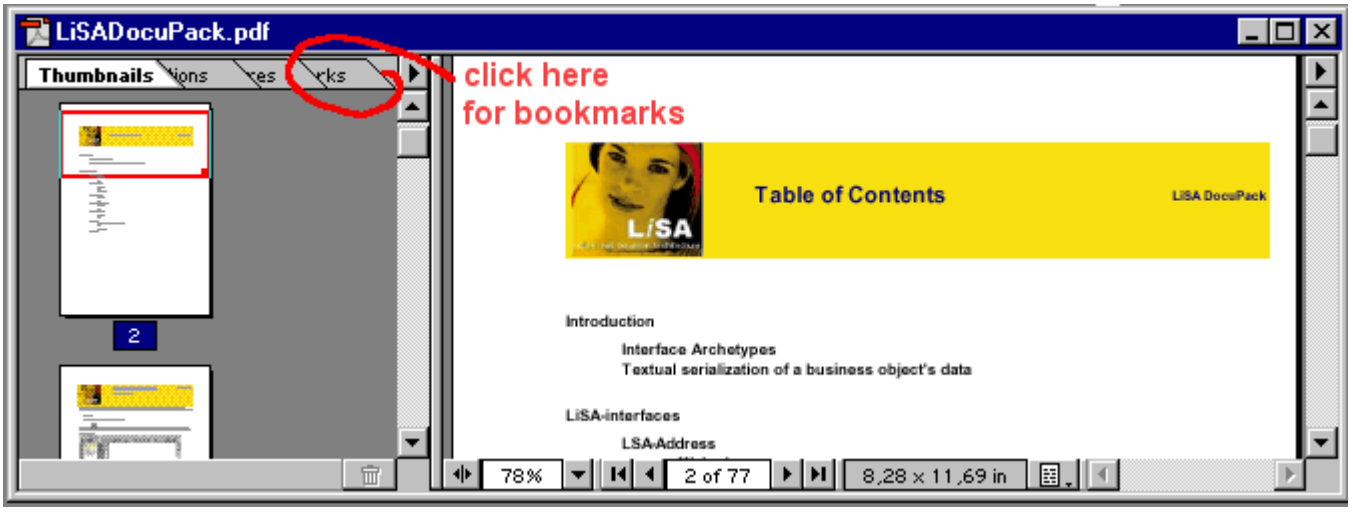
- **LSA-Address**
  - **Minimal**
  - **Full**
- **LSA-Archive**
  - **Minimal**
  - **Full**
  - **Advanced**
- **LSA-Glossary**
  - **Minimal**
  - **Full**
- **LSA-Organization**
  - **Minimal**
  - **Full**
- **LSA-Configure**
  - **Minimal**
  - **Full**
- **LSA-Help**
  - **Ready to use design blocks**
  - **Full**
  - **Advanced**
- **LSA-UDF**

Document: Navigation in the PDF file

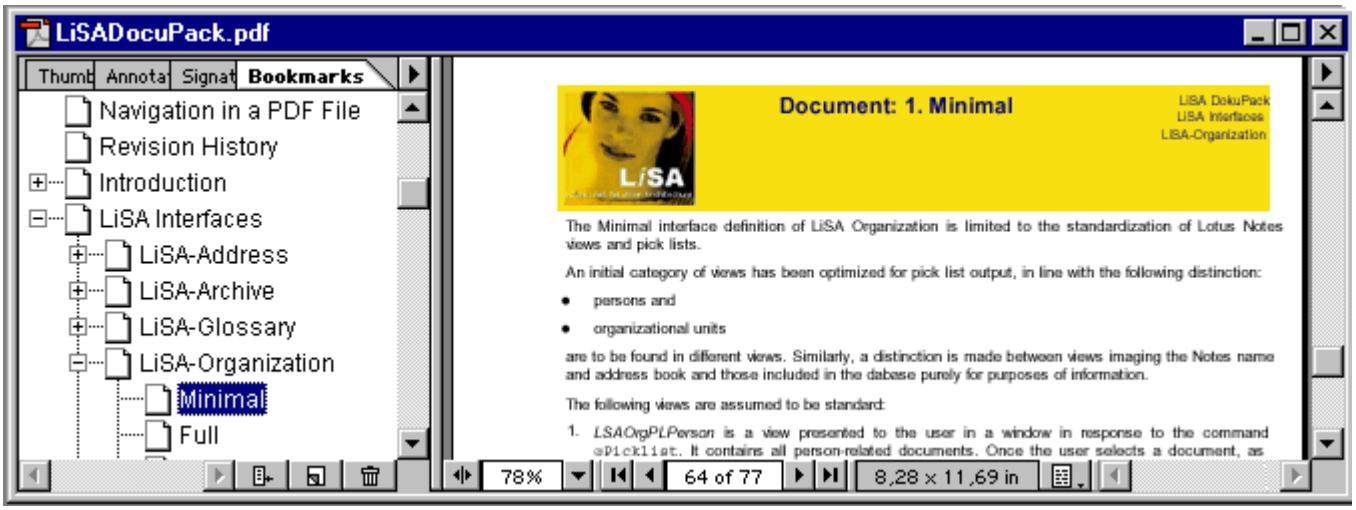
The LiSAInterface Description is also offered in the PDF format of Adobe Acrobat 4.0. To support navigation through this vast document, this file has been set up to permit movement to individual pages

- directly
- with the aid of chapter bookmarks

Press the  icon in the menu bar or from teh menu window->Show Thumbnails F6 option to obtain a view of the pages. A list of miniaturized pages then appears. A click on one of the page symbols on the left-hand side is sufficient to choose the page you want to go to.



Press the tab to obtain a list of chapters. It is then possible to move to a specific chapter by clicking on the table of contents that then appears in this case minimal of LiSA-Organization. By clicking on the + / - symbol, it is possible to display or collapse the sub-headings. For instance in the screenshot below, the pages below LiSA-Archive is collapsed while LiSA Address is expanded.





Document: Revision History

With this release 2.1 of LiSA we incorporate various small suggestions from the field into the LiSA Organization interface definition:

- With the furher evolution of Traits (dynamic and relational traits) it has become aparent that it is nolonger appropriate to define that on minimal level. Trait interfaces will be defined and extended on the Full level.
- The programatic access to traits has been improved.

Some of these changes are already implemented in this LiSA 2.0e most are a statement of future direction.

With the previous release LSA 2.0 the support for the i-net impacted the implementation has been introduced. In order to stay upward compatible we use LSA for internal design elements and file naming rather than LiSA! The changes in the Organization interface announces with 1.1 have become effective. Some additions have been made in the following parts:

- LiSA-Organization Advanced interface is extended in the area of Traits to allow relational and dynamic traits to be accessed.
- LiSA-Help Reusable blocks have changed due to the additional implementation of the contextsensitive WEB help.
- LiSA-Configure the LSALuDataByKey view has changed due to the additional design alias (as opposed to the replica alias)

With the previous release LSA 1.1 we have extended the interface due to experience in lage installations. You will find changes in following parts:

- LSA-Organization Minimal and Full. The structure <*LotusU*> will change in the future due to limitationsin Notes and these changes are incorporated allready. This version will remain upward compatible to 1.0 but will change in the next version. It is strongly recommendet to use the new conventions.

Those parts that will be discontinued are underlined and those parts **new are bold** font. There will be explanations like

HINT
explaining the reason for the change.



Chapter: Introduction

The Lotus i-net Solution Architecture (LiSA) interface recommendations have been drawn up with the goals of ensuring that :

- applications can find each other and exchange data without problems
- applications are easy to configure
- implementation-neutral interfaces are defined
- frequently used data objects can be updated
- examples and a reference implementation are available

This LiSA-Interface Reference Manual is a description of the LiSA interfaces. A database in an application can :

- provide interfaces
- use interfaces

Three levels of interface usage are specified in order to facilitate description. Not all of the levels are consistently defined for all LiSA interface areas. The following levels :

- minimal,
- full,
- advanced.

This manual defines the LiSA-Interface for various areas. A database can provide and use one or more interfaces at the same time. LiSA-Address, for example, provides an interface to access contact or organization data, but at the same time uses LiSA-Configure and LiSA-Glossary. LiSA-Address does not use the LiSA-Archive interface which is fine as long as no conflicting interfaces in that area exists. The current areas, as provided in this manual, may be supplemented by other areas as standardisation requires in the future. When using the term LiSA *fitting,* it is our goal to make such interdependencies easy to identify.

A database is LiSA 'fit' if :

1. it uses LiSA-Configure to achieve installation independence, define parameters and declare interfaces,
2. for all areas where an LiSA-Interface defined in this manual applies

- it is committed which level is provided by the implementation of the LiSA-Interface definition
- the level used is specified and not compromised by alternative ways of accessing the data

The compatibility chart below shows how these interrelations apply.

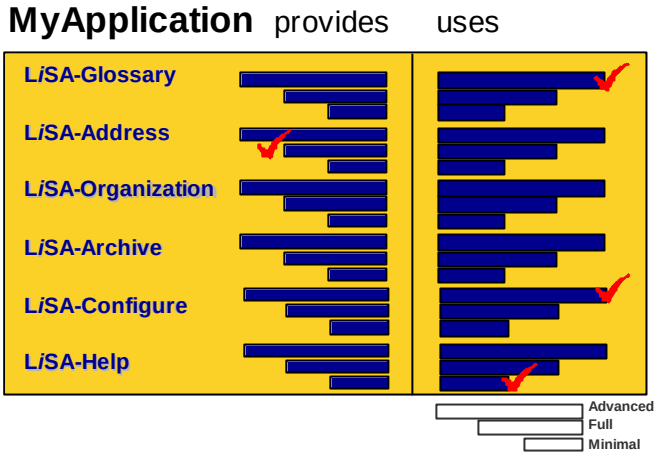


Fig.: LiSA fitting, graphic representation

This makes it very easy for the purchaser of a solution to check whether the solution is compatible with the applicable application environment.

The present paper defines the interfaces for each of the LiSA modules, and describes them in detail at the same time. For upward compatibility the internal names stay LSAxxx.

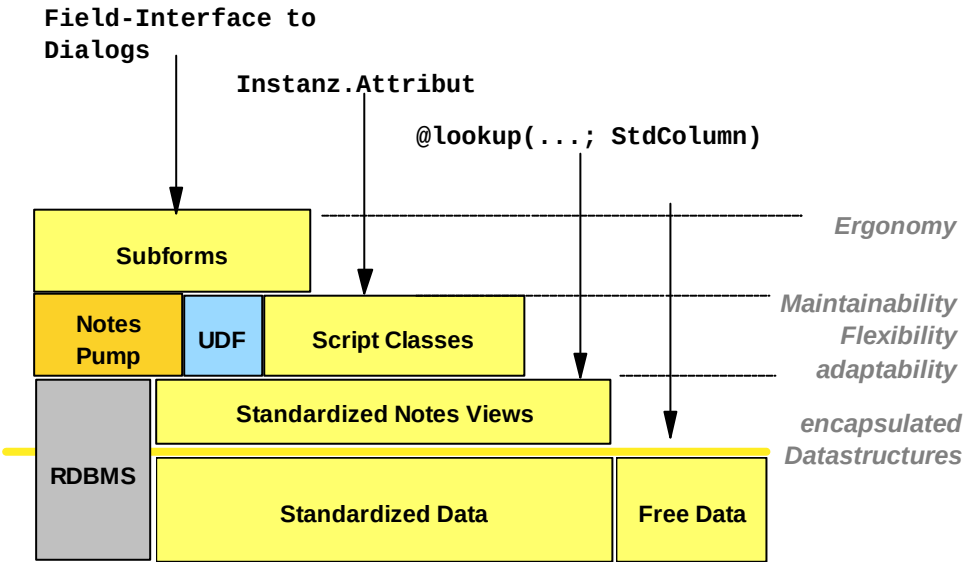
LiSA itself provides the interfaces in the sense of a reference, and with the LiSA-Integrator's Tutorial offers a series of completed design blocks that mount on the LiSA interfaces. This makes it easier for developers to integrate into their applications functionalities based on the LiSA standard.

Document: Interface archetypes

On what levels are interfaces offered?

The Lotus Notes development platform offers a versatile range of programming options. LiSA must offer developers interfaces for the various programming variants, whereby the Notes-specific features must be afforded the appropriate degree of consideration. However, it is not just a question of programming interfaces. The increasing degree of solution- orientation is shifting the focus from developer to end user, and in consequence towards the user interface to be provided. This leads to the development of interface levels mounting on each other, and to interface archetypes that exist on each of these levels.

As a basic rule, the process described above involves a move away from concrete data fields in favor of encapsulated business objects. Examples of business objects filed in Notes might be organizational units, telephone numbers, persons, address labels, keywords, document history, and so forth. The interfaces have been defined with the goal of maximizing the re-usability of these business objects. Figure 1 (below) shows the various interface levels together with the possible interface archetypes.



(Fig.1: Interfaces in dependence on solution orientation)

Since re-usability and the associated cost reductions represent merely one of the quality attributes aspired to by the definition and implementation of interfaces, other aspects must be granted equal consideration. Alongside the uniformity and ergonomy of the user interface, decisive factors are flexibility and enhancability in line with the wishes of the customer. In view of the diverse programming possibilities of Notes, the interface must, on the one hand, leave open the range of options that cover anything from simple "plug & play" to the specific integration into customized modules. In particular, it is vital to ensure that interfaces possess the degree of stability necessary to guarantee the sustained interoperation of those applications mounted on them and likewise of the interfaces provided. The method by which this is accomplished is demonstrated below for the various levels on the basis of interface archetypes.

Interface archetypes

Let us begin with the **Data Level** of a business object filed in Lotus Notes in a number of fields. The primary concern at this level is to find appropriate attributes for a business object, and to define the interface in line with these attributes . One approach to defining an interface might be to agree upon field names. However, the disadvantage of this method is that it is very rigid and permits neither modifications of any kind nor customer-specific extensions. For these reasons, it is adopted only in three cases for which Notes offers no alternative options:

1. In connection with free data whose data structure is unknown, meaning it is merely possible to standardize the data to a transfer field with fixed name and pre-determined structure. Here it is a question of implementing the maximum degree of data independence from various partners and customers.
2. When transferring data to dialog boxes in which special transfer fields serve to parametrize the dialog or act as return fields which adopt the results. On quitting the dialog, these fields are removed by default and are not integrated in the data model.
3. Many fields are pre-calculated which are to be located by means of a search formula. These fields are part of the standard definition.

To enhance performance in views, it is becoming customary to use pre-calculated fields with keywords, categories, content information for external view displays, and selection lists for users. For this purpose, LiSA will file the business objects as serial information in fields. However, we will refrain from incorporating fields of this kind into the interface definition. Since the field names have proved to be unsuitable for standardizing the data, it is a matter of finding a mode of representation which is

- self-describing,
- field-independent,
- sufficiently simple with regard to usage of various programming languages, and
- capable of incorporating complex structures.

This mode of representation is used in all cases where the data of a business object are being filed or transferred, and where exclusively Lotus Notes is used for this purpose, one example being views. We have termed this 'serialized, textual representation'.

Standardization at **Views level** aims to ensure that business objects can be simply located while at the same time access is provided to their data. A basic distinction between views is whether they are used by users or programs. For this reason, it is essential that business objects possess an external identifier which is, if possible, unique for the user, and also an unique internal access key for the programs. Care must be taken that the external representation is concise and comprehensible, and above all is used consistently. Business objects therefore frequently offer fields in which these identifiers are pre-calculated.

As a rule, access to the data of an object is carried out in several steps:

1. The business object is initially located with the aid of specific criteria.
2. The internal key, the external identifier and structured basic data are provided.
3. The internal key is used to access further structured data.

For instance, a person is located by surname and first name and initial of the second name. At the same time, the most important information -- such as pre-defined telephone number, address, etc. -- is provided. In the case of ambiguities, it is possible in this way to identify and select the required person. The unique key is thus obtained, and is used to obtain all of the person's phone numbers, which for their part each have uniquely assigned identification terms such as secretarial office, mobile, private.

Another case for the usage of views is related to creating the basis for editing relations. In an object hierarchy, for example, it is interesting to know which objects are beneath the relevant object, and which ones are higher. Although it is possible to provide this information in views, only programs are in a position to interpret the information.

The level described in the following can be described as the **Action Level** of a business object. It builds on the data levels lying below it. Such actions might be comparatively simple, like for example the

interpretation of a standardized data structure using a function library of LotusScript functions, but might equally be as complex as the detection of a database on the basis of its alias. Another variant is the provision of LotusScript classes providing both the attribute access and also the methods. Where this is insufficient, modules mounting on the Notes-Domino API are provided. Often the functions serve the purpose of breaking up relations or updating information maintained as redundant. In all events, the goal is to offer shared functionality in easily serviceable form. In particular, this encompasses

- documentation of the principle sequences so that their deployment is possible,
- no direct, unencapsulated dependencies on the fields,
- adjustability of the concrete environment only via parameters,
- re-usability and capsulation in design elements or modules.

At this point it becomes clear, if indeed it was not already so, that Notes programming does not permit capsulation with pre-specified criteria but that dependencies arise on the destination environment used. Furthermore, the many fields that interact are frequently motivated by technical considerations or layout requirements.

The last level -- the **Subform Level** -- therefore functions as a clasp around the various design elements, and in many cases can be interpreted as a business object. Subforms can also be used as dialog boxes, and then act as the user interface of specific functions. Subforms thus gather together aspects such as

- ergonomic layout,
- integration into the environment and concretization of general functions,
- provision of business objects as modules for solutions.

In cases where a subform is a business object in its own right, it carries this object's data and makes it available to the other interfaces. If a subform is merely a dialog interface, the data are called up and provided/returned as separate interface fields which are not, however, part of the data model. Once the data has been called up and copied, it is deleted. Temporary fields only exist for the duration of the dialog.

Since subforms work upon the assumption that specific views, configuration parameters or other design elements are available, these call-up requirements are documented in full.

Interfaces Levels acc. to LiSA Standard

The second level (Views) is generally defined as Minimal or Full Standard. This makes it possible to carry out simple data access operations to any desired implementations of the LSA interfaces, including to the description of the serialized text format of an object.

The first level (Data) is generally first added in Full Standard, since if possible no implementation dependencies such as field names ought to be required.

The third level (Actions) is frequently a component of the Advanced Standard, in which specific function libraries are defined. In the case of UDF functions for LiSA Configure, the interface is already located at Full Level.

The fourth level (Design Blocks) falls under the category usage of LiSA standards, and is therefore part of the LiSA Developer Pack but not an interface standard of its own. LiSA Help is an exception; in it, specific design blocks are necessary for functional capability, and the interfaces are described in this paper.



Document: Textual serialization of a business object's data

The concept of data provision

In LiSA, the data of an object are presented in a form described as textually serialized. Various field contents are combined to form a string of characters, and can be easily called up in views. Textual serialization complies with the following rules:

Rule 1:

The data string begins with a delimiter which is used to divide the string into data blocks.

As a rule, LiSA uses the pipe symbol (vertical line "|") as delimiter at the top level in the hierarchy. A serialized data string thus has the following structure:

|<Data block 1>|<Data block 2>|<Data block 3>|...|<Data block_n>

Rule 2:

The first data block contains any required number of additional delimiters enabling the follow-on data blocks to be hierarchically structured.

Further delimiters used by LiSA are, as a rule, the characters "#" and "~". In such cases, <Data block 1> then reads "#~", and the data string has the following form:

|#~|<Data block 2>|<Data block 3>|...|<Data block_n>

Rule 3:

The folllow-on data blocks contain data which is subdivided through the specified delimiters.

If the delimiters "#" and "~" are used, then a single data block invariably has the following format:

<Data_0_0>~<Data_0_1>~...~<Data_0_n_0>#<Data_1_0>~...~<Data_1_n_1>#...#<Data_m_0>~...~<Data_m_n_m>

As is clear from this example, the delimiters employed must not appear in the various data values, since they would alter the structure of the string. The programmer must therefore ensure that these values do not appear in data.

Rule4:

Data included in data blocks must not include either the first delimiter (Rule 1) or any of the delimiters specified in Data Block 1 (Rule 2).

In order to make more flexible the somewhat rigid data structure for the position parameters, the following convention applies.

Rule 5:

Data occupying position 0 in a data block can be interpreted as the identifier for the data block.

This makes it possible to identify data not only on the bais of their position in the data string but also by a designator (identifier). This identifier identifies a data area in an object, e.g. such data as is customer- or partner-specific. Examples of LiSA standard identifiers, which are described in detail in the next chapters, are listed below:

1. LotusC Contact in LiSA Address (Contact)
2. LotusO Organization in LiSA Address (Organization)
3. LotusP Telephone number in LiSA Address (Phone)
4. LotusW Web site in LiSA Address
5. LotusA Address information in LiSA Address
6. LotusL Label in LiSA Address (Label)
7. LotusT Type definition in LiSA Organization
8. LotusU Organizational unit in LiSA Organization (Unit)

   ...

This means the following serial character strings are semantic equivalents:

|#~|LotusC#<Data_1_1>#...|LotusO#<Data_2_1>...

|#~|LotusO#<Data_1_1>#...|LotusO#<Data_2_1>...

The serial text format is thus a very simple method of providing structured data. As the following examples show, it has a versatile range of deployment, and offers the following particular advantages:

- Flexibility in choice of applications, e.g. various address data fields with identical interface
- Openness for integration of free data (e.g. customer or partner data)
- Relatively simple to process both with Notes @ formulas or LotusScript
- Diverse range of possible usages

Examples of formulas of this kind:

a simple list of the months of the year:	~~Jan~Feb~Mar~Apr~May~Jun~Jul~Aug~Sep~Oct~Nov~Dec
a corporate hierarchy:	||ACME|Geschäftsbereich Vertieb|Region Nord|Büro Hamburg
a list of couples with telephone numbers:	~=~Mobile=+49 123 1234-56~Office=+49 456 1111- 11~...
a tuple list with personal data:	^~=^Michaela^Muster^Dr.^Mobile=+49 123 123-45~Office=+49 456 456- 78^CLIENT DATA^...
a CGI parameter list:	&&Martin&Mustermann&Dr.&Musterweg+2&12345+Musterberg

Usage of data blocks

Our example is based on the following text format representing a very general case:

```
|#~|A#B~C~D#E#F|G#H#J|K#L~M~N
```

The following levels can be detected on the basis of the nested text data:

```
1                   ->  delimiter
2                   ->  A#B~C~D#E#F
2.0                 ->  A
2.1                 ->  B~C~D
2.1.0               ->     B
2.1.1               ->     C
2.1.2               ->     D
2.2                 ->  E
2.3                 ->  F
3                   -> G#H#J
3.0                 ->  G
3.1                 ->  H
3.2                 ->  J
4                   -> K#L~M~N
4.0                 ->   K
4.1                 ->   L~M~N
4.1.0               ->     L
4.1.1               ->     M
4.1.2               ->     N
```

With **Position 0** being interpreted as **identifiers**, the following descriptions might for example be accomplished:

```
2.0     -> <A>
2.1.0   -> <A>.<B>
3.0     -> <G>
4.0     -> <K>
4.1.0   -> <K>.<L>
```

Thus, the data elements are accessible both by name and by their position:

```
2.1.1   -> <A>.<B>.1 ->     C
2.1.2   -> <A>.1.2   ->     D
3.1     -> <G>.1     ->     H
```

In order to further heighten independence and enhance processibility, each of the data elements can be assigned a name. Since we are dealing with business objects, the names can also be understood as attributes, and assign to the latter their position:

```
Adam        <A>
Berti       <A>.1.0
Claudia     <A>.1.1
Dora        <A>.1.2
Eva         <A>.2
Fritz       <A>.3
Gustav      <G>
Hannelore   <G>.1
```

```
Jörg        <G>.2
Karl        <K>
Ludger      <K>.<L>
Mike        <K>.<L>.1
Nicole      <K>.<L>.2
```

This means attributes can now be addressed by name.

```
Adam        ---> A#B~C~D#E#F
Gustav      ---> G#H#J
Jörg        ---> J
Ludger      ---> L~M~N
Nicole      ---> N
```

LiSA standardizes its business objects with the aid of this mode of representation. As a rule, this mapping is filed as a parameter in LiSA Configure, meaning it can be easily added to without any changes having to be made to the program code.

Access to <A> as an @ formula produces the result "B~C~D#E#F", and might have the following appearance:

```
_S0:=@Left(Data;1);
_SubSeparators:=@Word(Data;_S0;2);
_S1:=@Left(_SubSeparators;1);
_S2:=@Middle(_SubSeparators;1;1);

_DataA:=@Middle(Data;_S0+"A"+_S1;_S0);
```

or if <A>.2, i.e. "E"

```
_DataA2:=@Word(_DataA;_s1;2);
```

The list <A>.1 is edited to "B":"C":"D" as

```
_DataListA1:=@Explode(@Word(_DataA;_s1;1);_S2);
```

Caution:

@explode("a#b###e#f";"#") produces the list
"a":"b":"e":"f"
and not the result that might conceivably be expected and returned by the Script LiSA function VExplode
"":"":"a":"b":"":"":"e":"f"
This means extreme caution is therefore advised when using @explode. One method of avoiding this problem is to previously fill up the blank entries by adding a random character, in this case the main delimiter as this is for sure not in the string, producing the following re-arrangement

```
"ab###e#f"                    becomes with
@ReplaceSubstring(...;"#";"|#")
"ab|#|#|#e|#f"                becomes with a final "|" added
"a|#b|#|#e|#f|"               becomes with @explode(...;"#")
"a|":"b|":"|":"|":"e|":"f|"   becomes with @leftback(...;1)
"a":"b":"":"":"e":"f"
```

Therefore the following code is more robust

```
_DataListA1:=@leftback(
                @Explode(
                  @ReplaceSubstring(
                    @Word(_DataA;_s1;1);
                    _s2;
                    _s0+_s2)+_s0;
                   _S2);
                1);
```

The LiSA Developer Pack offers enhanced functions in LotusScript such as:

```
CreateDatalist(Data as String,Datalist List as Node)
CreateMapping(Attributelist as Variant, Datalist List as
Node)
GetValue(Query as String, DataList List as Node)
```

but also simple string functions:

```
VExplode (Data as String, DelimChar as String)
Vtrim (Data as String)
VLeft (Data as String, Delim as Variant)  Delim maybe
String or number
etc.
```

Integration of partner applications or customer data

The data of various partners now has to be transferred to a field such as `ClientData` , which is defined by name in the interface of the module. Merely the delimiters are specified in this process. The submask determines one or more random names for the data blocks and places together these subblocks. The module now integrates these data into the serialized textual object data. The field might, for example, contain the blocks <P> and <S>:

```
_Data=|#~|P#Q#R|S#T~U#V~W
```

Interface documentation

In the desciption of each LiSA Interface, the serial data sructure is described in a table under **Data format of the business objects**, e.g.

Identifier	Position in data string	Explanation
LotusX	<LotusX>.0	Xxxx Data of a business object
	<LotusX>.1	What is in this position?
	...	

The data string which is stored in the field does not always have an identical number of data blocks. This is due to the fact that not every document must necessarily include all the data blocks described, as well as to the possibility of additional data blocks being added to the string.



Document: View Design Philosophy

The design goals for the LiSA view structures were to:

- Minimise the required number of views;
- Allow the maximum performance to be gained by retrieving the appropriate amount of data in one lookup;
- Minimise the overall amount of data stored by views;
- Encapsulation of the data into views so other applications do not need to understand implementation details.

These objectives are sometimes in conflict with one another. i.e. a view may have the same information as another for performance requirements. In other places there are cascaded lookups necessary to finally retrieve the required data.

The LiSA-Minimal interface provides generally applicable functionality that can be easily implemented by other applications with minimal impact. This permits existing applications to become LiSA conformal and serve as data sources for LiSA conformal applications.

The LiSA-Full interface exposes specialised functionality that is found in the LiSA reference implementation. Adding this functionality may require modification to existing applications. e.g. adding the reserved field name used by LiSA for searching.

There are currently five styles of views used within the LiSA-Generic modules:

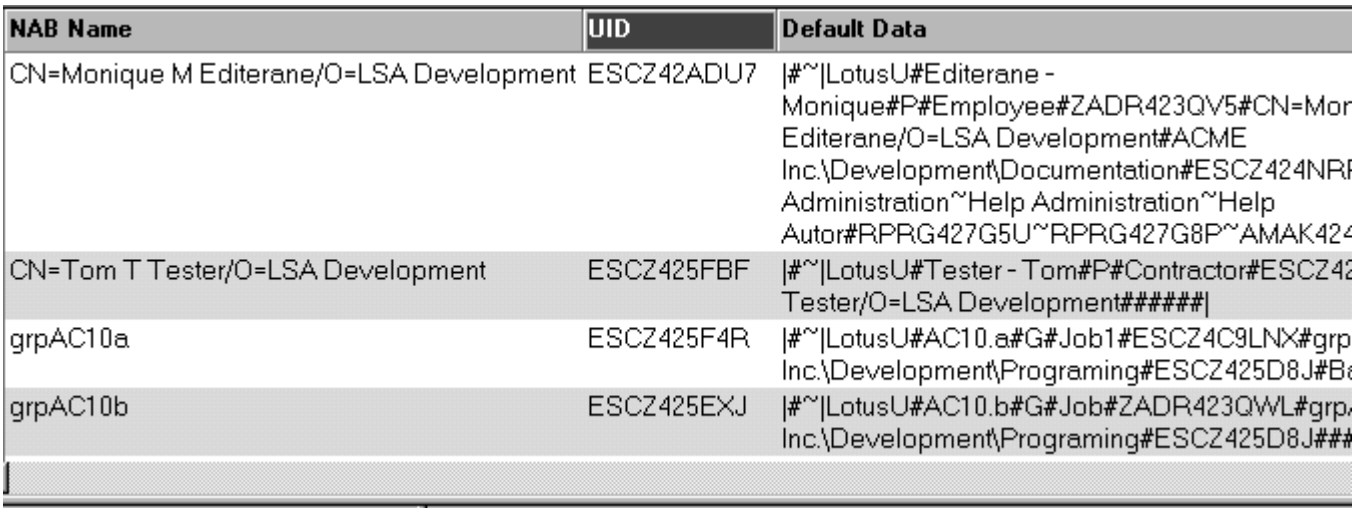
- 'access and Redirect';
- 'Pick and Redirect';
- 'Primary Access';
- 'Extended Access';
- 'Hierarchy Navigation'.

All of these views have been conceived with a design philosophy that provides a consistent set of data for both user-driven selection dialogs and application code.

Access and Redirect Style

This stype of views is designed for use by programs rather than direct navigation by users. These programs usually have some kind of external link as a starting point such as the @username, a key from an external system, the document unique id or alike.

The first column contains this external link, which typically is a unique value, although this is not a requirement. The second column is an internal key from the document. The third column usually contains default information in serial data format, which is purely for performance reasons.



NAB Name	UID	Default Data
CN=Monique M Editerane/O=LSA Development	ESCZ42ADU7	|#~|LotusU#Editerane - Monique#P#Employee#ZADR423QV5#CN=Mor Editerane/O=LSA Development#ACME Inc.\Development\Documentation#ESCZ424NRF Administration~Help Administration~Help Autor#RPRG427G5U~RPRG427G8P~AMAK424
CN=Tom T Tester/O=LSA Development	ESCZ425FBF	|#~|LotusU#Tester - Tom#P#Contractor#ESCZ42 Tester/O=LSA Development######|
grpAC10a	ESCZ425F4R	|#~|LotusU#AC10.a#G#Job1#ESCZ4C9LNX#grp Inc.\Development\Programing#ESCZ425D8J#Ba
grpAC10b	ESCZ425EXJ	|#~|LotusU#AC10.b#G#Job#ZADR423QWL#grpA Inc.\Development\Programing#ESCZ425D8J###

An application can use one of these views by performing a @DbLookup with a key to find the required entity. The application can extract the default information from the third column. Where more information is needed the *internal key* from the second column is used as the argument to a second @DbLookup operation performed on one of the views in 'Extended Lookup' style (in list above and described in detail below).

These views contain the letters LU (Lookup) after the module name.

e.g. `LSAAddrLUnnnnnn`

Pick and Redirect Style

These views are designed for use by applications using the @Picklist command. This means they are used by humans and may differ in terms of the presentation and data. The common approach to unify the use of these different views is to hide the first column which contains the *internal key* of the document. Any number of additional columns can be added to this style of view according to application requirements.

These views are named on the basis of how the additional columns have been arranged. An example in LiSA-Address is:

```
96. LSA-Minimal\3. LSAAddrPLOrganization
```

This view is used as the source of data for the Organisation dialog

InternalKey	Organisation
	ACME Inc.
ESCZ424NVA	ACME Inc.
	Board of directors
ESCZ424NMP	Board of directors
	Controlling
ESCZ424NKN	Controlling
	AC65.f
	AC65.f
	Information Technology
ESCZ424NHM	Information Technology
	AC53.a
	AC53.a
	Development
ESCZ425CTQ	Development
	Documentation
ESCZ424NRR	Documentation
	Programing
ESCZ425D8J	Programing

These views contain the letters PL (Pick List) after the module name.

e.g. LSAAddrPLnnnnnn

Primary Access Style

There are usually some primary access views, that hold all of the application data so that any application that holds the internal key can successfully retrieve the required data. A simple example the 'Pick List' view transforms the user's selection into the internal key that is used in the second lookup.

UID	DocID	Default Data	DefaultHierarchy
AMAK424MZM	C125678C004CDB40C12566BE	|#~|LotusU#Help Autor#C#Competence#ESCZ424LJE#LSA	Help Autor
AMAK424P6H	C125678C004CDB40C12566BE	|#~|LotusU#Organization Master Access#C#Competence#ESCZ424LJE#L -	Organization Master Access
AMAK424P7H	C125678C004CDB40C12566BE	|#~|LotusU#Organization Administration#C#Competence#ESCZ424 Ressources#LSAA424MWH~AMAK424P.	Organization Administration
AMAK424P8Q	C125678C004CDB40C12566BE	|#~|LotusU#Organization Owner#C#Competence#ESCZ424LJE#LS Ressources#AMAK424PAK##|	Organization Owner
AMAK424P9T	C125678C004CDB40C12566BE	|#~|LotusU#Organization Creator#C#Competence#ESCZ424LJE#L Ressources~Sales#AMAK424PAK~ESC	Organization Creator
AMAK424PAK	C125678C004CDB40C12566BE	|#~|LotusU#Human Ressources#C#Competence	Human Ressources

Usually they have the internal key in the first column and the Document Unique ID in the second column. the following columns provide serial Data and if necessary other keys or external names of the object. As this kind of view is really universal it shows all documents.

`LSAxxxLUAllByID`

But why using an internal key rather than stick with the Document Unique ID ? The answer is, that the Document Unique ID is not consistent across non-replica copies. This consistency is a requirement for referential purposes, so an internal key is required. The Document Unique ID is still useful where you wish to open a document by programmatically controlling the user interface. This is the purpose of the

`LSAxxxLUAllByUNID`

view.

Extended Lookup Style

An 'Extended Lookup' view is again intended for use by programs. The first column contains the *internal key* from the document, and may appear in the view more that once. The second column is the Unique ID of the document. The third column contains the data of interest in serial data format.

InternalKey	DocumentUniqueID	PhoneList
ESCZ424LQJ	C125678C004CDB40C12566BE003CAEBF	~~x~Mobil~49~171~33333~~+49 - (0) 171 - 333
ESCZ424MBX	C125678C004CDB40C12566BE003CAEC0	~~x~intern~~~~19~ - 19
ESCZ424MC4	C125678C004CDB40C12566BE003CAEC1	~~x~intern~~~~13~ - 13
ESCZ424MC4	C125678C004CDB40C12566BE003CAEC1	~~~Mobil~49~173~22222~~+49 - (0) 173 - 222 2
ESCZ424MC7	C125678C004CDB40C12566BE003CAEC3	~~x~intern~~~~15~ - 15
ESCZ424MC8	C125678C004CDB40C12566BE003CAEC4	~~x~intern~~~~18~ - 18
ESCZ424MTX	C125678C004CDB40C12566BE003CAEC7	
ESCZ424MWP	C125678C004CDB40C12566BE003CAEC9	~~x~Fax~49~40~345672~~+49 - (0) 40 - 345 672
ESCZ424MWP	C125678C004CDB40C12566BE003CAEC9	~~~Zentrale~49~40~345670~~+49 - (0) 40 - 345
ESCZ424N9Z	C125678C004CDB40C12566BE003CAECA	
ESCZ424NHM	C125678C004CDB40C12566BE003CAECD	
ESCZ424NKN	C125678C004CDB40C12566BE003CAECE	
ESCZ424NMP	C125678C004CDB40C12566BE003CAECF	
ESCZ424NRR	C125678C004CDB40C12566BE003CAED0	
ESCZ424NVA	C125678C004CDB40C12566BE003CAED1	~~x~Zentrale~49~211~12345~0~+49 - (0) 211 -

This structure allows encapsulation of the data for a given entity independently of whether the data should be sourced from a single document or multiple documents in the database. This provides for the implementation of a one to many relationship. This flexibility also allows us to overcome the data size limits of Notes views by limiting the amount of data required to be present at any point in the view.

These views are usually named on the basis of the type of data being returned by the third column. Two examples from LiSA-Address are:

`97. LSA-Full\2. LSAAddrLUPhoneSByID` for telephone numbers.

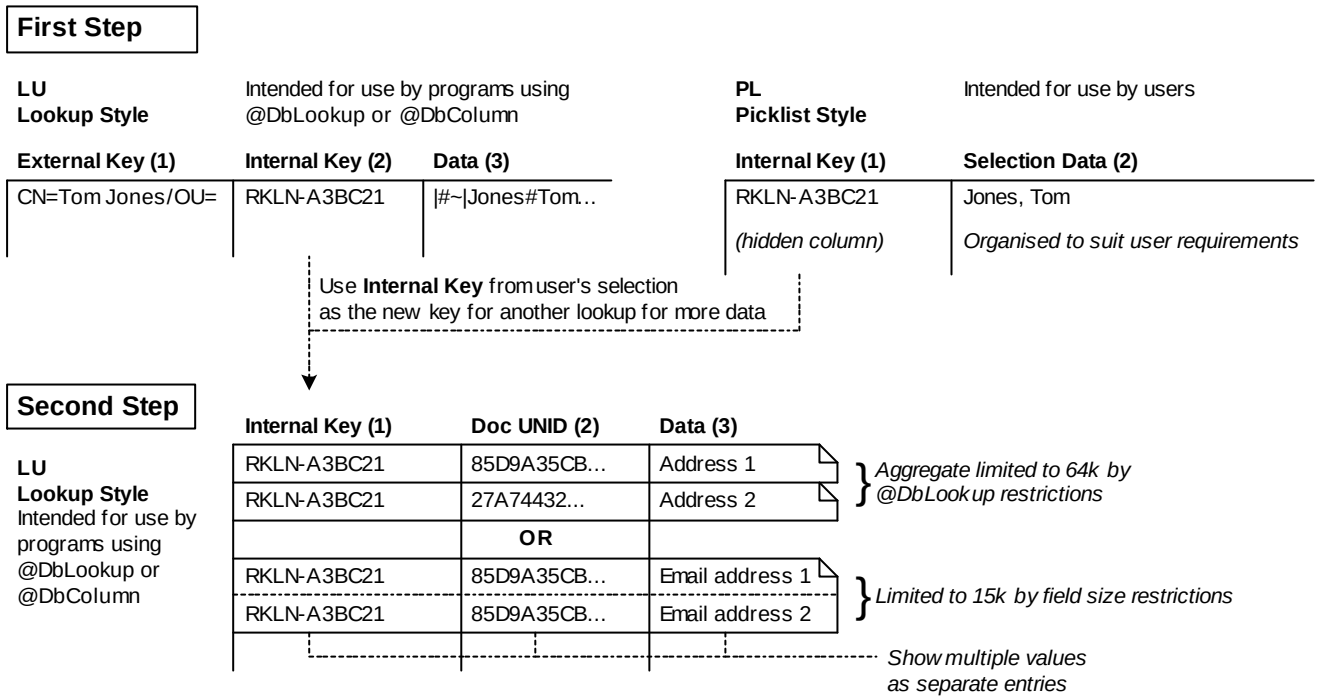
`97. LSA-Full\2. LSAAddrLUEMailSByID` for e-mail addresses.

These views also contain the letters LU after the module name.

e.g. `LSAAddrLUnnnnnn`

Cascaded Access To Data Entities

The following diagram illustrates the relationship between the LU and PL views found in an LiSA-Minimal implementation and the LU views found in the LiSA-Full implementation.



Hierarchy Navigation Style

A 'Hierarchy Navigation' view is designed to allow navigation within a hierarchy. This means that recursive lookups are performed down the hierarchy until the user has made a final selection. The first column is a concatenation of the hierarchy level and the data that describes the current node.The second column contains the data that describes the children nodes. This data is interpreted to present a set choices for user selection.

LSAHierarchyLevels	LSAHierarchyNode
0|~	1999/02/17 12:43:00~city guide~Stadtführer
0|~	1999/02/17 14:52:38~vehicle~Fahrzeug
0|~	1999/06/11 06:30:45~earth~Erde
0|~	1999/09/27 13:19:59~Cooperative~Genossensch
1|~|1999/02/17 12:43:00~city guide~Stadtführer	1999/02/18 00:19:22~accommodations~Unterkür
1|~|1999/02/17 12:43:00~city guide~Stadtführer	1999/02/18 00:19:35~gastronomy~Gastronomie
1|~|1999/02/17 12:43:00~city guide~Stadtführer	1999/02/18 00:19:48~culture~Kultur
1|~|1999/02/17 12:43:00~city guide~Stadtführer	1999/02/18 00:19:58~the sights of town~Sehensw
1|~|1999/02/17 14:52:38~vehicle~Fahrzeug	1999/02/28 10:03:41~types~Typen
1|~|1999/06/11 06:30:45~earth~Erde	1999/02/18 00:12:56~Europe~Europa
1|~|1999/06/11 06:30:45~earth~Erde	1999/02/18 00:13:14~America~Amerika
1|~|1999/06/11 06:30:45~earth~Erde	1999/06/11 06:33:35~Asia Pacific~Asiuen Pazifik
2|~|1999/02/17 12:43:00~city guide~Stadtführer|199!	1999/02/11 15:11:40~hotels~Hotels
2|~|1999/02/17 12:43:00~city guide~Stadtführer|199!	1999/02/17 13:54:50~camping site~Campingplät
2|~|1999/02/17 12:43:00~city guide~Stadtführer|199!	1999/02/17 13:57:45~youth hostels~Jugendherbe

In operation an application performs a @DbLookup with a key of "0|~" using the second column as the return data. This data is processed to present a list of choices for user selection. After the user selects one of the choices the application pre-pends the next level to the selected choice and performs another @DbLookup to return the list of choices for this new level. This process continues until there are no more levels and the user selects a document or cancels the selection operation.

The Hierarchy Navigation style is found in LiSA-Orga and LiSA-Glossary.

Traversing The Hierarchy Navigation Style View

LSA Minimal / Full

LU
Lookup Style
Intended for use by programs using @DbLookup or @DbColumn

Prepend next level to the data of the current selection and perform a lookup to retrieve the next set of choices

Key (1)	Data (2)
0|~	1999/02/01 10:00:00~Friends~Freunde
0|~	1999/02/01 11:00:00~Job~Beruf
1|~1999/02/01 10:00:00~Friends~...	1999/02/01 10:00:25~Club~Verein
1|~1999/02/01 10:00:00~Friends~...	1999/02/01 10:00:12~School~Schule
1|~1999/02/01 11:00:00~Job~Ber...	1999/02/01 11:00:23~Colleagues~Kollege
1|~1999/02/01 11:00:00~Job~Ber...	1999/02/01 11:00:23~Clients~Kunden

Hierarchy Levels Modelled By This Example

Level 0	Level 1
Friends	
	Club
	School
Job	
	Colleagues
	Clients

Creating your own views in LiSA Style

If you want to design your own views in an LiSA style, you should follow the following steps.

1. Define the Business objects that you want to be reflected in the view, their attributes and make sure they have
   - unique internal key
   - appropriate external representation
   - well defined the data blocks of serialized textual representation
2. Determine their relation to existing objects. You will make up
   - a main document

     If the object is not related to any other element or the relation is maintained in data
   - a subpart within existing documents

     if exactly one object is directly related to another object.
   - A response document

     If the object contains restricted information with restricted access rights, e.g. A person is associated a response containing salary, bonusplan etc.

     If there may be multiple instances related to a main object, e.g. memos to a contact.

3. Define access and redirect views based on the external representations, which will lead you to the primary access views

4. Define Pick and redirect if necessary

5. If they are main documents, add the Objects into the primary access views

```
LUAllByID
LUAllByUNID
```

6. If the information is related to existing objects both as subpart or as response, create your own extended access



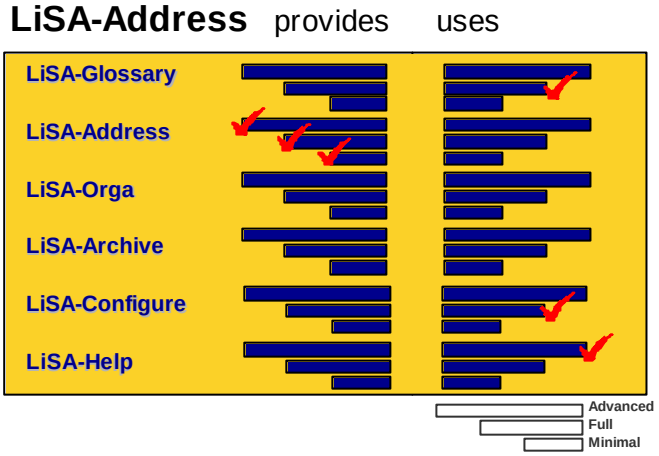
Chapter: LiSA Interfaces

The LiSA interfaces describe and allow access to data, which is usable within any kind of solution.

The interface definition is based on the assumption that it is should be possible to replace the Lotus reference implementations by alternative implementations. Therefor the interface definitions of level LSA-Minimal were defined in a manner, that by integration of some standardized views an application can become compliant.. Domino implementation require the use of Fields in some areas. The Full interface extends the minimal requirements in by also standartizing few fieldnames to assure conformality. The Full interface also reals with 1:n relations in that this multiple data access is made available via the interfaces. The Advanced interfaces targets to features of the LiSA implementation such as script interfaces and alike.

Chapter: LiSA-Address

LiSA Address offers interfaces at the Minimal, Full and Advanced levels and mounting on the interfaces of components LiSA Configure and LiSA Help. LiSA Address is a reference implementaton of the address management interfaces documented within the scope of this docu-pack.





Document: 1. Minimal

The minimal interface definition of LiSA Address is limited to the standardization of Lotus Notes views. The following views are required as standard:

1. *LSAAddrLUOrganizationByName* is conceived as view for the command `@DBLookup` and the methods `GetDocumentByKey` of the `Notesview` Notes class. All documents and organizational units containing address information appear in this view. The first column is ordered and contains the key for locating a document. The content of this column is standardized and contains the hierarchical organization name as a serial text string in which hierarchical levels are separated by the pipe symbol ("|"). Columns 2 and 3 are results columns.

   - Note: in LotusScript, certain versions of Notes react erroneously to "\" in the key column. The serial LiSA format is therefore used instead.

2. *LSAAddrLUContactByName* is conceived as view for the command `@DBLookup` and the methods `GetDocumentByKey` of the `Notesview` Notes class. All person-related documents containing address information appear in this view. The first column is ordered and contains the key for locating a document. The content of this column is standardized in the form "<Surname> <Forename> <Initial of 2nd name>". Columns 2 and 3 are results columns.

3. *LSAAddrPLOrganization* is conceived as a view presented in a window to the user in response to the command t `@Picklist`. The view contains all person-related documents that store address information. As the result of a selection by the user, the first column should be returned (concealed, unsorted) containing the unique key by which the address information of the selected documents can be read in the view *LSAAddLUAllByID.* The information for the user is clearly hierarchically ordered in the follow-on columns.

4. *LSAAddrPLContact* is a view presented in a window to the user in response to the command t `@Picklist`. The view contains all person-related documents that store address information. As the result of a selection by the user, the first column should be returned (concealed, unsorted) containing the unique key by which the address information of the selected documents can be read in the view *LSAAddrLUAIIByID.* The information for the user is clearly alphabetically ordered in the follow-on columns. .

5. *LSAAddrLuAllByID* is conceived as view for the command `@DBLookup` and the methods `GetDocumentByKey` of the `Notesview` Notes category. All person-related and organizational unit documents containing address information appear in this view. The first column is ordered and contains a unique key for accessing the address information of the documents. Column 2 is a results column.

6. *LSAAddrLUIdByDocUNID* is conceived as view for the command `@DBLookup` and the methods `GetDocumentByKey` of the `Notesview` Notes category. All person-related and organizational unit documents containing address information appear in this view. The first column is ordered and contains a document unique ID for accessing the address information of the documents. Column 2 is a results column.

No standardization of fields has been undertaken, due primarily to the following two considerations.

- It is desirable to keep the minimal interface as narrow as possible in order to facilitate the integration of existing applications into the LiSA Standard.
- Due to the combined calculation of data in one interface field of a document, replication and storage conflicts almost inevitably arise when changes are simultaneously made to one document by two people or when changes are made from a decentral location. These conflicts cannot be automatically solved by the "Merge" option offered by Notes.

In order to avoid restricting the capabilities of Lotus Notes in this regard, the existence of merely a small number of views can be guaranteed at minimal level. This limited number of fields unfortunately means that the full bandwidth of the search functionality available in Notes is not available. Although in a full-text indexed database the option still exists of obtaining certain documents by search command, this method cannot use the possibilities of a field-name independent search. For these reasons, certain fields are standardized in the Full interface definition.

The views are named with the aliases stated above. We recommend that the views be grouped with an external name, so that the following picture results:

96. LSA-Minimal
    1. LSAAddrLUOrganizationByName
    2. LSAAddrLUContactByName
    3. LSAAddrPLOrganization
    4. LSAAddrPLContact
    5. LSAAddrLUAllByID
    6. LSAAddrLUIDByDocUNID

Views

Column	Title	Example	Description
LSAAddrLUOrganizationByName			
1	Name	||Cybertech Corporation|Development|Sales	Full hierarchical name of an organizational unit. Hierarchic levels are specified as a serial text string delimited by "|" (pipe)
2	Internal Key	SK45KN14	Internal key for data access
3	Defaultdata	|#~|LotusO#Cybertech Corporation\Development#Cybertech Corporation#Development###Computer Industry|LotusP#+49 - (0) 89 - 595 940 - 12|LotusE#|LotusW#http://www.cybertech.de/development|LotusA#Office Munich####Maiselfinger Allee#12##82332#Munich#Bavaria#Germany|LotusL#Cybertech Corporation#Maiselfinger Allee 12 # #82332 Munich, Bavaria#Germany#####|	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" and "LotusL". See description "Data Format of Business Objects", below, for definition of the data blocks.
LSAAddrLUContactByName			
1	Name	Groß Franz P	The name of the contact is chosen as key in the form "<Surname> <Forename> <Initial of 2nd forename>". If no forename is used, only the surname (without appended blank) is displayed. If there is no 2nd name initial, only the surname and forename are displayed, separated by a blank (with no appended blank after the forename).
2	Internal Key	SK45KN14	Internal key for data access.
3	Defaultdata	|#~|LotusC#Franz#Groß#P#Dr.###M###CEO|LotusO#Cybertech Corporation#Cybertech Corporation####|LotusP#+49 - (0) 89 - 595 940 - 1|LotusE#FGross@cybertech.de|LotusW#|LotusA#Office Munich####Maiselfinger Allee#12##82332#Munich#Bavaria#Germany|LotusL#Cybertech Corporation#Herrn Dr. Franz Groß#Maiselfinger Allee 12 # #82332 Munich, Bavaria#Germany####||	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" and "LotusL". See description "Data Format of Business Objects", below, for definition of the data blocks.
LSAAddrPLOrganization			
1	Internal Key (concealed)	SK45KN14	Internal key for data access
2	arbitrary, e.g. name	Cybertech\Sales	This view is conceived for usage in pick lists. The unique key for the first column should be returned as result. The displayed name is freely definable.
3	arbitrary	arbitrary	
LSAAddrPLContact			
1	Internal Key (concealed)	SK45KN14	Internal key for data access
2	arbitrary, e.g. name	Groß Franz	This view is conceived for usage in pick lists. The unique key for the first column should be returned as result. The displayed name is freely definable.
3	arbitrary	arbitrary	
LSAAddrLUAllByID			
1	Internal Key	SK45KN14	Internal key for data access

2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUniqueID
3	DefaulData	|#~|LotusC#Franz#Groß#P#Dr.###M###CEO|LotusO#Cybertech Corporation#Cybertech Corporation####|LotusP#+49 - (0) 89 - 595 940 - 1|LotusE#FGross@cybertech.de|LotusW#|LotusA#Office Munich####Maiselfinger Allee#12##82332#Munich#Bavaria#Germany|LotusL#Cybertech Corporation#Herrn Dr. Franz Groß#Maiselfinger Allee 12 # #82332 Munich, Bavaria#Germany####||	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" and "LotusL". See description "Data Format of Business Objects", below, for definition of the data blocks.
LSAAddrLUIdByDocUNID			
1	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUniqueID as key for data access
2	Internal Key	SK45KN14	Internal key
3	DefaulData	|#~|LotusC#Franz#Groß#P#Dr.###M###CEO|LotusO#Cybertech Corporation#Cybertech Corporation####|LotusP#+49 - (0) 89 - 595 940 - 1|LotusE#FGross@cybertech.de|LotusW#|LotusA#Office Munich####Maiselfinger Allee#12##82332#Munich#Bavaria#Germany|LotusL#Cybertech Corporation#Herrn Dr. Franz Groß#Maiselfinger Allee 12 # #82332 Munich, Bavaria#Germany####||	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" and "LotusL". See description "Data Format of Business Objects", below, for definition of the data blocks.

Data format of the business objects

Business object <LSA SerialDataName>	Data block. position	Attributes
LiSA Address		
LotusC	<LotusC>.0	DATA BLOCK DESCRIPTION: person-specific data of a contact
	<LotusC>.1	Forename
	<LotusC>.2	Surname
	<LotusC>.3	Initial of 2nd name
	<LotusC>.4	Title
	<LotusC>.5	Suffix
	<LotusC>.6	Nickname
	<LotusC>.7	Gender
	<LotusC>.8	Date of birth
	<LotusC>.9	Functional task
	<LotusC>.10	Job description
LotusO	<LotusO>.0	DATA BLOCK DESCRIPTION: Organization-specific data
	<LotusO>.1	Organization identifier (full hierarchy)
	<LotusO>.2	Company (first element in hierarchy)
	<LotusO>.3	Department (last element in hierarchy)
	<LotusO>.4	Abbreviation
	<LotusO>.5	Synonym
	<LotusO>.6	Branch
LotusP	<LotusP>.0	DATA BLOCK DESCRIPTION:Standard telephone number
	<LotusP>.1	Telephone number
LotusE	<LotusE>.0	DATA BLOCK DESCRIPTION:Standard E-mail address
	<LotusE>.1	E-mail address
LotusW	<LotusW>.0	DATA BLOCK DESCRIPTION:Standard Web site
	<LotusW>.1	Url of Web site
LotusA	<LotusA>.0	DATA BLOCK DESCRIPTION:Postal address
	<LotusA>.1	Postal address
	<LotusA>.2	Care of addressee
	<LotusA>.3	Building
	<LotusA>.4	Room no.
	<LotusA>.5	Street
	<LotusA>.6	Number
	<LotusA>.7	Postbox
	<LotusA>.8	Postal code
	<LotusA>.9	Town
	<LotusA>.10	Country
	<LotusA>.11	Federal state
LotusL	<LotusL>.0	DATA BLOCK DESCRIPTION:Correspondence data
	<LotusL>.1	Line 1 of label
	<LotusL>.2	Line 2 of label
	<LotusL>.3	Line 3 of label
	<LotusL>.4	Line 4 of label
	<LotusL>.5	Line 5 of label
	<LotusL>.6	Line 6 of label
	<LotusL>.7	Line 7 of label
	<LotusL>.8	Line 8 of label
	<LotusL>.9	Letter opened
	<LotusL>.10	Letter sealed

The minimal interface definition of LiSA Address standardizes views in order to guarantee uniform access to the data in an address database. The full interface defintion further pursues this concept by adding views permitting access to multiple addresses, telephone numbers, e-mail addresses and Web sites of an addressee. The individual views are as follow:

1. *LSAAddrLUPhoneSByID* is conceived as a view for the command `@DBLookup` and the methods `GetDocumentByKey` in the `Notesview` Notes class. All documents containing telephone numbers appear in this view. The first column is ordered, and contains a unique key for accessing the telephone numbers of a person or organizational unit. The second column delivers the DocumentUniqueID of the document storing the corresponding telephone number(s) of the addressee. The third column is a results column.

2. *LSAAddrLUEMailSByID* is conceived as a view for the command `@DBLookup` and the methods `GetDocumentByKey` in the `Notesview` Notes category. All documents containing mail addresses appear in this view. The first column is ordered, and contains a unique key for accessing the e-mail addresses of a person or organizational unit. The second column delivers the DocumentUniqueID of the document storing the corresponding e-mail address(es) of the addressee. The third column is a results column.

3. *LSAAddrLUWebPageSByID* is conceived as a view for the command `@DBLookup` and the methods `GetDocumentByKey` in the `Notesview` Notes category. All documents containing Web page information appear in this view. The first column is ordered, and contains a unique key for accessing the Web pages of a person or organizational unit. The second column delivers the DocumentUniqueID of the document storing the corresponding Web page(s) of the addressee. The third column is a results column.

4. *LSAAddrLUAddresseSByID* is conceived as a view for the command `@DBLookup` and the methods `GetDocumentByKey` in the `Notesview` Notes category. All documents containing postal address information appear in this view. The first column is ordered, and contains a unique key for accessing the address information of a person or organizational unit. The second column delivers the DocumentUniqueID of the document storing the corresponding postal address(es) of the addressee. The third column is a results column.

5. *LSAAddrLUOrganizationSByID* is conceived as a view for the command `@DBLookup` and the methods `GetDocumentByKey` in the `Notesview` Notes category. All documents containing organizational units appear in this view. The first column is ordered, and contains a unique key for accessing those organizational units in which the person or organizational appears as a member. The second column delivers the DocumentUniqueID of the document storing the corresponding organizational unit (s) of the addressee. The third column is a results column, i.e. it contains the hierarchic names of the organizational units as they are presented in the view.

This interace definition, however, goes one step further by creating an advanced search interface via the standardization of two fields.

1. *LSAAddrIntOrganizationSearch* is used to locate organizational unit documents. The standard prescribes that the hierarchic names of those organizational units associated with the Notes document must be entered in this field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.

2. *LSAAddrIntContactSearch* is used to locate person-specific documents. the standard prescribes that the name of the person must be filed in the field in the following form: "<Surname> <Forename>" and "<Forename> <Surname>". In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.

What makes a standardization of fields expedient? Due to the Notes-specific restrictions on size, unlimited usage of the frequently employed commands `@DBLookUp` and `@DBColumn` to locate data and present results in fields and menus is not possible. In the case of a larger database, there are frequently only two options:

1. To generate LotusScript classes which assume a mode of "storage management" for views, generally through iterative operation. These approaches are often non-performing, and not suitable for standardization.
2. To use the search functionalities of Lotus Notes. In a Notes environment, this is the customary method and offers space for standardization.

In full-text indexed databases it has so far been possible to do without standardized fields. On the other hand, a simple full-text search often delivers a vast number of documents not required in the search context since certain (guide-text) fields with no causal connection to the search request have been searched. It is possible to search through a non-full-text indexed database with the "Search" method of the LotusScript class "NotesDatabase". The search is based on the syntax of the "SELECT" commands. In this connection, it has proved to be expedient to formulate the search request on the basis of the existence and content of fields.

The views are named with the aliases stated above. We recommend that the views be grouped with an external name, so that the following picture results:

```
97. LSA-Full
      1. LSAAddrLUPhoneSByID
      2. LSAAddrLUEMailSByID
      3. LSAAddrLUWebPageSByID
      4. LSAAddrLUAddresseSByID
      5. LSAAddrLUOrganizationSByID
```

Fields

Structure	Example	Description
LSAAddrIntOrganizationSearch (Text, multi value)		
"<Name of organizational unit 1>": "<Name of organizational unit 2>": ... "<Name of organizational unit n>": "<arbitrary string 1>": "<arbitrary string 2>": ... "<arbitrary string m>"	"Cybertech\Sales"	This field is used to locate organizational unit documents. The standard prescribes that the hierarchic names of those organizational units associated with the Notes document must be entered in this field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
LSAAddrIntContactSearch (Text, multi value)		
"<Surname> <Forename>": "<Forename> <Surname>": "<arbitrary string 1>": "<arbitrary string 2>": ... "<arbitrary string m>"	"Groß Franz": "Franz Groß": "Franz P Groß, Kundenr. 4771"	This field is used to locate person-specific documents. The standard prescribes that the hierarchic names of the person must be filed in the field in the formats *Surname Forename* and *Forename Surname*. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.

Views

Column	Title	Example	Description
LSAAddrLUPhoneSByID			
1	Internal Key	SK45KN14	Unique key for the document, allowing access to the data.
2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUnique ID of the document providing the information.
3	PhoneList	"~~~Office~44~2342~21312~32~+44 - (0) 234 2 - 213 12 - 32": "~~x~Privat~44~323~24321~~+44 - (0) 323 - 243 21"	Telephone number format complies with the description "LSAPhoneList" in "Data formats of the business objects" (below).
LSAAddrLUEMailSByID			
1	Internal Key	SK45KN14	Unique key for the document, allowing access to the data.
2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUnique ID of the document providing the information.
3	EMailList	"~~x~Notes~Michaela Kleinmann/cybertech/DE": "~~~Internet(privat)~M.Kleinmann@t-online.de"	E-mail address format complies with the description "LSAEMailList" in "Data formats of the business objects" (below).
LSAAddrLUWebpageSByID			
1	Internal Key	SK45KN14	Unique key for the document, allowing access to the data.
2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUnique ID of the document providing the information.
3	WebPageList	"~~x~Private Homepage~http://Mittermeier.t-online.de": "~~~Homepage~http://www.bwildschuetz.de"	Web page format complies with the description in "LSAWebPageList" in "Data formats of the business objects" (below).
LSAAddrLUAddresseSByID			
1	Internal Key	SK45KN14	Unique key for the document, allowing access to the data.
2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUnique ID of the document providing the information.
3	PostalAddressList	"##Privat####Hansemann Str.#324##89889#Munich#Bavaria#Germany"	Postal address format complies with the description in "LSAPostalAddressList" in "Data formats of the business objects" (below).
LSAAddrLUOrganizationsSByID			
1	Internal Key	SK45KN14	Unique key for the document, allowing access to the data.
2	DocumentUniqueID	B93B19B20DC2FF91C1256636005CAAD0	DocumentUnique ID of the document providing the information.
3	OrgUnitList	"ACME\Sales\International"	Hierarchic name of organizational unit as displayed.

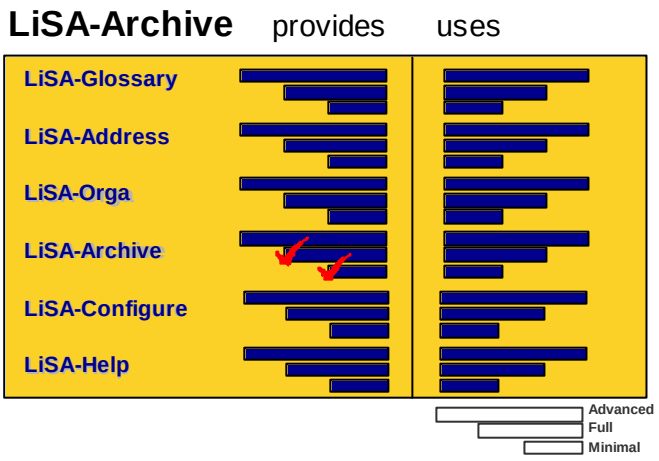
Data formats of the business objects

Business object <LSA SerialDataName>	Data block. position	Attributes
LiSA Address		
LSAPhoneList	2.0	Flag for specified number (x or blank)
	2.1	Description of telephone number
	2.2	Country code
	2.3	Area code
	2.4	Number
	2.5	Direct dial
	2.6	Formatted telephone number
	3. ...	...and so forth for every element in list
LSAEMailData	2.0	Flag for specified e-mail address (x or blank)
	2.1	Description of e-mail address
	2.2	E-mail address
	3. ...	...and so forth for every element in list
LSAWebPagedata	2.0	Flag for specified URL (x or blank)
	2.1	Description of Internet page
	2.2	URL
	3. ...	...and so forth for every element in list
LSAPostalAddressList	2.0	Address description
	2.1	Care of addressee
	2.2	Building
	2.3	Room
	2.4	Street
	2.5	Number
	2.6	Postbox
	2.7	Postal code
	2.8	City
	2.9	Country
	2.10	Federal State
	3...	...and so forth for every element in list

Chapter: LiSA-Archive

LiSA Archive offers interfaces at the levels Minimal and Full. These interfaces are designed to make it easier to equip with archiving functions Notes applications. The goal is the step-by-step neutralization of Notes-specific features in such a way that storage in any desired archive software and middleware is possible without previous knowledge of Notes, and thus administrable by the person in charge of the archive. On the other hand, it is aimed to enable Notes designers to provide their applications with archive interfaces without first having to gain detailed knowledge of archiving systems and methods.

The implementation is realized as a UDF module.





Document: 1. Minimal

The minimal interface of the LiSA Archive module provides the following functionality:

- Preparation for archiving of Lotus Notes documents
- Filing of documents in an archive system, including automatic keyword assignment
- Search through index system for archived documents
- Retrieval of located documents, and restoration as Lotus Notes document

1. Preparation for archiving of Lotus Notes documents

Lotus Notes documents can rarely be viewed as isolated objects. Instead, they "live" from their relationships to other documents, views, and databases. This embedding in a specific environment makes it difficult to archive Lotus Notes documents, since particularly in the case of long-term archival (> 1 year) it cannot be assumed that the environment existent when the document was archived will still exist or be restorable when the document is recovered from the archive.

An important step in the archiving process is therefore to liberate the document from all its relationships to other objects and make it available as, to some degree, a "photograph" in a Rich Text field of a new document. Since this field cannot provide structured information, specific, information-rich fields have to be copied to the new document. These fields can be calculated using Lotus Notes formulas.

The minimal interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAASetArcRules"; "RuleDBSvr":"RuleDBPath";
"ID");
@DbCommand("UDF" : "NoCache"; "@LSAAArchive"; "PREPARE");
```

2. Filing documents in an archive system

Any kind of Lotus Notes documents can be filed in an archive system via the LiSA Archive interface. Due to the reasons explained in 1.), above, some documents are not suitable for archiving, however, meaning the filing is often the second step after the preparation measures described above. At Minimal level, all information relating to the index and archive system (system, server, user, password, archive type, index type) are saved in a configuration file, e.g. in FOREVER.INI in the case of FOREVER. These parameters therefore do not require administration in the application. However, only a single archive system can be addressed at this level.

The minimal interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF" : "NoCache"; "@LSAASetArcRules";"RuleDBSvr":"RuleDBPath";
"ID");
@DbCommand("UDF" : "NoCache"; "@LSAAArchive"; "ARCHIVE");
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

3. Search for archived documents

This functionality is available if Lotus Notes documents have been transferred to an archive system. The applicable documents must first be looked for in the index system of the archive. The result of the search is presented in the form of Lotus Notes documents.

At Minimal level, all information relating to the index and archive system (system, server, user, password) are saved in a configuration file, e.g. in FOREVER.INI in the case of FOREVER. These parameters therefore do not require administration in the application.

The minimal interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF" : "NoCache"; "@LSAASetQuery"; "IdxField"; "Operator";
"Value"; "JoinOperator");
@DbCommand("UDF" : "NoCache";
```

```
"@LSAASetQueryRules";"RuleDBSvr":"RuleDBPath"; "ID");
@DbCommand("UDF" : "NoCache"; "@LSAAQuery");
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

4. Retrieval

This functionality allows Lotus Notes documents to be recovered from archived documents. Documents can be restored in any database in which the user is authorized to create documents. It is also necessary to ensure that the database contains the template with which the restored documents can be displayed. This rule does not apply if the template was saved in the document when archiving took place.

The minimal interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF" : "NoCache"; "@LSAASetRetSource"; "ArcID");
@DbCommand("UDF" : "NoCache"; "@LSAASetRetRules"; "RuleDBSvr" :
"RuleDBPath"; "ID");
@DbCommand("UDF" : "NoCache"; "@LSAARetrieve");
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

Interface to the rule database

The interface to the database which stores the rules is a view called LSAALURulesByID. It consists of two columns:

1. ConfigurationID (e.g. "23" or "MyDb.MyForm")
2. Parameter string with the following structure:

Nr.	Content
LSA-Archive: LSAALURulesByID	
<LotusS>.0	LotusS (=Lotus Source)
<LotusS>.1	SourceDB
<LotusS>.2	SourceSrv
<LotusS>.3	SelectionType (VIEW, FORMULA, UNID)
<LotusS>.4	Selection
<LotusS>.5	Response Hierarchy ("0"=ohne, "1"=mit)
<LotusS>.6	SourceForm
<LotusS>.7	ChangeField
<LotusS>.8	ChangeType ("N"=Names; "O"=Object; "NA"=Author; "NR"=Reader;"_"=None
<LotusS>.9	ChangeFormula
<LotusS>.10	ErrorFormula
<LotusS>.11	Changed documents ("1"=Main documents; "2"=all documents)
<LotusT>.0	LotusT (=Lotus Target DB)
<LotusT>.1	TargetDB
<LotusT>.2	TargetSrv
<LotusT>.3	TargetForm
<LotusT>.4	Target richttext field
<LotusT>.5	Compute with form flag ("0"=disabled; "1"=enabled)
<LotusT>.6.0	Field Source
<LotusT>.6.1	Field Target
<LotusT>.6.2	Field Type
<LotusT>.7.0	Field Source
<LotusT>.7....	Field Target...

Example:

^#~^LotusS#LSA\LSDARC.NSF##VIEW#Archiving#1##LSAArcFlag#_#"2"#"99"#2^LotusT#LSA\LSBARC.NSF##MainTopic#Body##Subject~Subject~#@Uppercase(Categories)~Categories~#From~From~N

Alongside LSA, the functionality of archiving, search and retrieval is based on a LiSA compliant archiv middleware like for example GROUP forever. A prior installation of an archive software is required for the archiving, search, and retrieval functionalities.

Restrictions
The following restrictions apply when using the minimal interface:

- only one archive system supported
- only one index system supported
- only one archive type supported
- only one index type supported

These four properties are stored in a file. In the case of GROUP forever, e.g., the file FOREVER.INI stores the following parameter:

```
[LSAArcMinimal]
ArcSystem=File
ArcServer=192.123.122.45
ArcUser=user
ArcPassword=password
ArcType=1
ArcBuffer=c:\forever\arcbuf
RetBuffer=c:\forever\retbuf

IdxSystem=ODBC
IdxServer=LSAArchIx
IdxUser=lsaa
IdxPassword=password
IdxType=LSADemo
```

The listed values are examples for the parameters which depend on the archive software.



Document: 2. Full

The full interface of the LiSA Archive module provides the following functionality:

- Preparation for archiving of Lotus Notes documents
- Filing of documents in an archive system, including automatic keyword assignment
- Search through index system for archived documents
- Retrieval of located documents, and restoration as Lotus Notes document

Differences to the minimal interface:

- The usage of more than one archive and index systems is possible.
- Usage of the configuration database is optional, meaning the individual parameters can also be dynamically calculated from within the application (e.g. index type, archive type, field mapping).

1. Preparation for archiving of Lotus Notes documents

Lotus Notes documents can rarely be viewed as isolated objects. Instead, they "live" from their relationships to other documents, views, and databases. This embedding in a specific environment makes it difficult to archive Lotus Notes documents, since particularly in the case of long-term archival (> 1 year) it cannot be assumed that the environment existent when the document was archived will still exist or be restorable when the document is recovered from the archive.

An important step in the archiving process is therefore to liberate the document from all its relationships to other objects and make it available as, to some degree, a "photograph" in a Rich Text field of a new document. Since this field cannot provide structured information, specific, information-rich fields have to be copied to the new document. These fields can be calculated using Lotus Notes formulas.

The full interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAASetArcRules"; "RuleDBSvr":"RuleDBPath";
"ID");
@DbCommand("UDF" : "NoCache"; "@LSAASetArcRenderRules"; "TargetRtField";
"Form");
@DbCommand("UDF" : "NoCache";
"@LSAASetArcSource";"SourceDBSvr":"SourceDBPath"; "SelectionType";
"Selection"; ResponseFlag);
@DbCommand("UDF" : "NoCache"; "@LSAASetArcSourceField"; "Formula";
"ErrorFormula"; "TargetField" : "Type");
@DbCommand("UDF" : "NoCache"; "@LSAASetArcTarget"; "TargetDBSrv" :
"TargetDB"; "TargetForm"; ComputeFlag);
@DbCommand("UDF" : "NoCache"; "@LSAASetArcTargetField"; "Formula";
"TargetField" : "Type");
@DbCommand("UDF" : "NoCache"; "@LSAAArchive"; "PREPARE");
```

2. Filing documents in an archive system

Any kind of Lotus Notes documents can be filed in an archive system via the LiSA Archive interface. Due to the reasons explained in 1.), above, some documents are not suitable for archiving, however, meaning the filing is often the second step after the preparation measures described above.

The full interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF":"NoCache"; "@LSAAArcRetSetParam"; "Parameter"; "Value")
@DbCommand("UDF":"NoCache"; "@LSAAArcRetOpenConnection"; "ArcSystem";
"ArcServer"; "ArcUser"; "ArcPassword")
@DbCommand("UDF":"NoCache"; "@LSAAIdxSetParam"; "Parameter"; "Value")
@DbCommand("UDF":"NoCache"; "@LSAAIdxOpenConnection"; "IdxSystem";
```

```
"IdxServer"; "IdxUser"; "IdxPassword")
@DbCommand("UDF" : "NoCache"; "@LSAASetArcRules";
"RuleDBSvr":"RuleDBPath"; "ID");
@DbCommand("UDF" : "NoCache"; "@LSAASetArcSource";
"SourceDBSvr":"SourceDBPath"; "SelectionType"; "Selection"; ResponseFlag);
@DbCommand("UDF" : "NoCache"; "@LSAASetArcSourceField"; "Value";
"ErrorValue"; "TargetField");
@DbCommand("UDF" : "NoCache"; "@LSAASetArcTargetField"; "SourceField";
"TargetIndexField");
@DbCommand("UDF" : "NoCache"; "@LSAAArchive"; "ARCHIVE"; "ArchiveType";
"IndexType");
@DbCommand("UDF":"NoCache"; "@LSAAIdxCloseConnection")
@DbCommand("UDF":"NoCache"; "@LSAAArcRetCloseConnection")
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

3. Search for archived documents

This functionality is available if Lotus Notes documents have been transferred to an archive system. The applicable documents must first be looked for in the index system of the archive. The result of the search is presented in the form of Lotus Notes documents.

The full interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF":"NoCache"; "@LSAAIdxSetParam"; "Parameter"; "Value")
@DbCommand("UDF":"NoCache"; "@LSAAIdxOpenConnection"; "IdxSystem";
"IdxServer"; "IdxUser"; "IdxPassword")
@DbCommand("UDF" : "NoCache"; "@LSAASetQuery"; "IdxField"; "Operator";
"Value"; "JoinOperator");
@DbCommand("UDF" : "NoCache"; "@LSAASetQueryRules";
"RuleDBSvr":"RuleDBPath"; "ID");
@DbCommand("UDF" : "NoCache"; "@LSAASetQueryTarget";
"TargetDBSvr":"TargetDBPath"; "TargetForm"; ComputeFlag);
@DbCommand("UDF" : "NoCache"; "@LSAASetQueryTargetField"; "IndexField";
"TargetField"; "TargetFieldType");
@DbCommand("UDF" : "NoCache"; "@LSAAQuery"; "IndexType");
@DbCommand("UDF":"NoCache"; "@LSAAIdxCloseConnection")
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

4. Retrieval

This functionality allows Lotus Notes documents to be recovered from archived documents. Documents can be restored in any database in which the user is authorized to create documents. It is also necessary to ensure that the database contains the template with which the restored documents can be displayed. This rule does not apply if the template was saved in the document when archiving took place.

The full interface provides the following functions for this functionality:

```
@DbCommand("UDF" : "NoCache"; "@LSAAInit");
@DbCommand("UDF":"NoCache"; "@LSAAArcRetSetParam"; "Parameter"; "Value")
@DbCommand("UDF":"NoCache"; "@LSAAArcRetOpenConnection"; "ArcSystem";
"ArcServer"; "ArcUser"; "ArcPassword")
@DbCommand("UDF" : "NoCache"; "@LSAASetRetSource"; "ArcID"; "ArchiveType");
@DbCommand("UDF" : "NoCache"; "@LSAASetRetRules"; "RuleDBSvr" :
"RuleDBPath"; "ID");
@DbCommand("UDF" : "NoCache"; "@LSAASetRetTarget";
"TargetDBSvr":"TargetDBPath"; ResponseFlag);
@DbCommand("UDF" : "NoCache"; "@LSAASetRetTargetField"; "Formula";
"TargetField" : "TargetFieldType");
@DbCommand("UDF" : "NoCache"; "@LSAARetrieve");
@DbCommand("UDF":"NoCache"; "@LSAAArcRetCloseConnection")
```

```
@DbCommand("UDF" : "NoCache"; "@LSAATerminate");
```

Restrictions

Alongside LSA, the prior installation of an LSA-conformal archive middleware, e.g. the GROUP forever product, is required for the archiving, search, and retrieval functionalities.

LiSA also provides for the GROUP forever product an enhanced UDF interface which must, however, be viewed for the time being as a proprietary extension of the LSA Standard. At present, LiSA does not standardize an advanced interface for archives.

The following additional functionality is conceivable for archive systems:

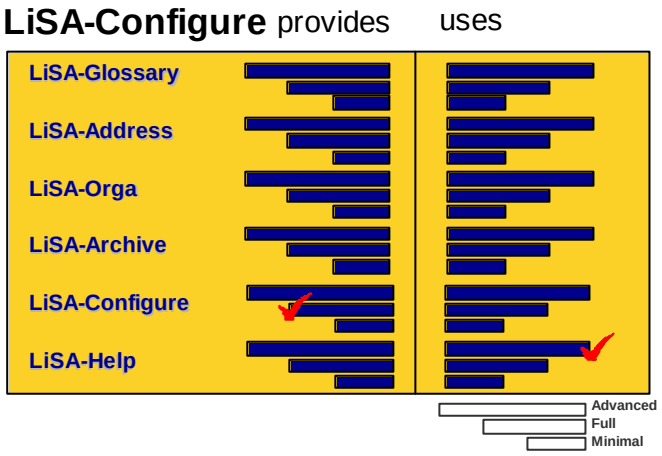
- Archiving and retrieval of any desired document
- Imaging (scanning, image file display, format conversion)

Chapter: LiSA-Configure

LiSA Configure is a kernel module. For that reason, no detailed description will be made of the Minimal level; the corresponding document merely explains the view used by the associated functions of the LiSA Library UDF for accessing the data of LiSA Configure. **We urgently advise you not to re-implement LiSA Configure but to use the original LiSA Configure module instead; this will guarantee compatibility with the full functional scope.**

LiSA Configure uses the LiSA Library UDF from the LiSA Developer Pack to maintain maximum interface speed and provide the @command language and LotusScript. These functions are therefore listed in this document as Full Interface. See LiSA Library UDF Help for a full description.

At the Advanced level, LiSA Configure for its part uses LSA Help.





Document: 1. Minimal

LiSA Configure is a kernel module whose usage is essential as a general rule. Via the view described in the following, it works together with the LiSA Library UDF, and in this respect acts as the minimal interface to LiSA Configure. This information is necessary only for troubleshooting.

We urgently advise you not to re-implement LiSA Configure but to use the original LiSA Configure module instead; this will guarantee compatibility with the full functional scope.

Views

No.	Title	Example	Description
LSALuDataByKey			
1	UDF key	e.g. A_P~@CONFIG~lsacfg.nsf~*~* I_A~LSAADDR P~3~*~LSLFOOT~*~*~LSADbOptions	Display of a document key according to the following scheme: for replica documents: T/A_ID~<ReplicaID> **DA_A~<DbAlias>** for file documents a list including: P_A~<DbAlias>~<TeamAlias>~<Location> P_I~<Interfaces>~<TeamAlias>~<Lokation> I_A~<DbAlias> A_P~<Server>~<Path>~<TeamAlias>~<Location> **DA_A~<DbAlias>** for location documents: SL_NL~<NABLocation> for team documents: TP_TA~<TeamAlias> for parameter documents: P~<Scope>~<Interface>~**<Alias>** ~<TeamAlias>~<Location>~<ParameterName> **Hint:** **<Alias> is the** • **DesignAlias for parameter with scope 7 (Design)** • **DbAlias for parameter with scope 3 (File) and 5 (Replica)** note: - non-assigned elements (except <Server>) must be filled with * - in path names, \ must be replaced by / (applies to search key only)
2	Data	@CONFIG,lsahelp.nsf,LSAHELP	Data according to following scheme: for file documents a list including: <Server> : <Pfad> : <DbTitle> for location documents a list including:

			<Location> : <Server> for team documents: <Path> for parameter documents: <Parameter value> for replica documents: <TeamAlias>
3	**DesignAlias**	LSACFG	the **DesignAlias** is displayed for file and replica documents
4	Cluster	1	1 is displayed only if "Allow Cluster" is selected in a file document, otherwise nothing is displayed
5	Interfaces	LSA ADDRESS	the list of assigned interfaces is shown for file documents
6	**DbAlias**	**LSAAGLOS**	**the DbAlias is displayed for file and replica documents**



Document: 2. Full

The Full interface uses LiSA Configure to save the configuration data. For access to data of LiSA Configure, functions of the LiSA Library UDF are used exclusively. These functions are described further below, together with examples.

We urgently advise that every LSA-compatible application provides the following document types so that administrators can install the application appropriately:

- Interface
- Template
- Scenario

in each case with

- Parameters.

It is equally recommended that the parameter

- LSADbOptions

be used consistently to conceal parts of the template if the following condition is fulfilled

```
!@IsMember("Show";LSADbOptions)
```

The LiSA Developer Pack contains a suitable common field.

Libraries

Generally, LiSA Configure must be accessed more than once in order to locate the path to a database, the alias of a database, or a parameter. A grounded knowledge of the corresponding search sequences is necessary in order to program the single access operations in @command language or Script. In order to be able to program an application to be largely independent of these search sequences, functions of the LiSA Library UDF are used to access LiSA Configure data. The table below provides an overview of these functions:

Function	Description
LiSA Library UDF	
PathByAlias	determines database path for a given database alias
AliasByPath	determines the alias of a given database
PathByInterface	determines for a given interface the paths and titles of the databases supporting the interface
Interface	determines all interfaces supported by a given database
CodeParam	allows parameter values that can consist of different data types to be converted into an internal format with which the values can be displayed as a text string
DecodeParam	converts parameter values from internal text display into corresponding Notes data format
SetParam	session parameter (value is saved fro duration of a Notes session) and cross-session paramter (in Notes.ini) can be set with this function; higher-range functions are administered in LiSA Configure
GetParam	reads a parameter value from LiSA Configure, Notes.ini, or a session parameter; the value is delivered in internal format and can be converted to the original data type with DecodeParam

Example of callup in @command language

```
_udf     := "UDF":"NoCache";

REM "Find a path to an alias";
_alias   := "DBALIAS";
_path    := @DbCommand(_udf; "@PathByAlias"; _alias);
@If(@IsError(_path);
        @Return(
           @Prompt([OK]; "Error";
                   "Path could not be found"));
        void
);

REM "Read a parameter value and decode it";
_name    := "LSADbOptions";
_coded   := @DbCommand(_udf; "@GetParam";_name;0);
_column1 := @DbCommand(_udf; "@DecodeParam";_coded;1);
_column2 := @DbCommand(_udf; "@DecodeParam";_coded;2);


REM "Code a parameter value and write it to Notes.ini";

_new     := @DbCommand(_udf; "@CodeParam";_column1;_column2);
_ret     := @DbCommand(_udf; "@SetParam";_name;2;_new);
```

Example of callup in LotusScript

```
Sub Click(Source As Button)
     Dim session As New NotesSession
     Dim db As NotesDatabase

     Set db  = session.CurrentDatabase

     Dim u As UDF
     Set u   = New UDF(db.Server,db.FilePath)
     On Error Goto handle_error

'    Find a path to an alias
     alias$  = "DBALIAS"
     ret     = u.PathByAlias(alias$)

'    Read a parameter value and decode it
     name$   = "LSADbOptions"
     coded   = u.GetParam(name$,0)
     decoded = u.DecodeParam(coded)

'    Code a parameter value and write it to Notes.ini

     new$    = u.CodeParam(decoded)
     ret     = u.SetParam(name$,2,new$)

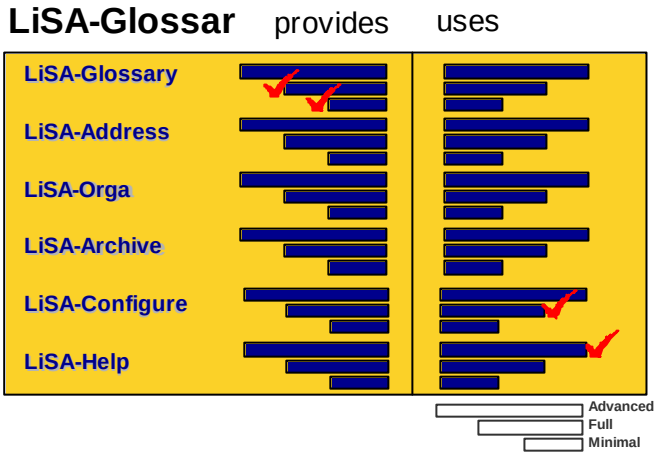
     Exit Sub

handle_error:
```

```
     Messagebox "Error" & Str(Err) & ": " & Error$,,"UDF"
     Resume Next
End Sub
```

Chapter: LiSA-Glossary

LiSA Glossary offers interfaces at the Minimal and Full levels, and mounting on the components LiSA Configure and LiSA Help. The LiSA Glossary is a reference implementation of the interfaces for keywording documented in the framework of this Docu-Pack.





Document: 1. Minimal

The minimal interface definition of LiSA Glossary is confined to the standardization of Lotus Notes views. The following views are required as standard:

1. *LSALuByHierarchy* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. In this view, all valid keywords are listed in their fully hierarchical order. Theis a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. In this view, all valid keywords are listed in their fully hierarchical order. The first column is sorted and contains the keyword path and actual keyword, the second column contains the DocumentUniqueID, and the theird column contains the HierarchyViewIndex, which is a simple view of the keyword in the first language. This view is mainly activated by the dialog boxes (`LSADlgSelectSeeAlsoTuple`, `LSADlgSelectSeeAlsoTupleGlossary`, ...).
2. *SALuByLevel* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. In this view, all valid keywords are listed according to their position in the hierarchic order. The first column is sorted, and contains the level occupied by the keyword within the hierarchy, linked with the keyword path. This means access is possible to all keywords on a specific level and in a specific branch. This view is used by dialog boxes, for instance, when selecting keywords (`LSADlgSelectSeeAlsoTuple`, `LSADlgEdit HierarchyTuple`, ...).
3. *LSALuLog* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. The view is the internal version of the keyword protocol in which all alterations to the keyword hierarchy are noted. The first column of the view contains a full, hierarchic representation of the old keyword, and the second column contains the new keyword, again in its full hierarchic representation. This view is deployed for updating keywords.
4. *LSALuByIndex* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. Although it corresponds to the view *LSALuByHierarchy*, the first and third columns have been swapped over, with the result that the simple display of the keyword (one language, no time stamp) gives information about the full pathname of the keyword. This view is used to re-synchronize with the Glossary keywords that are no longer available in the Glossary but were used to assign keywords to documents.
5. *LSAPLByHierarchy*is a view presented to the user in a window in response to the command `@Picklist`. The view contains a hierarchic, categorized display of all the terms in the glossary. As the result of a user selection, the first column (concealed, unsorted) is returned containing the unique key by which further information can be retrieved in the view *LSALuByHierarchy*and also allowing a document to be opened. The follow-on columns prepare the information for the user.
6. *LSAPLAlphabetic* is a view presented to the user in a window in response to the command `@Picklist`. The view contains an alphabeticall ordered list of all the terms in the glossary. As the result of a user selection, the first column (concealed, unsorted) is returned containing the unique key by which further information can be retrieved in the view *LSALuByHierarchy*and also allowing a document to be opened. The follow-on columns prepare the information for the user.

No standardization of fields has been undertaken, due primarily to the following two considerations.

- It is desirable to keep the minimal interface as narrow as possible in order to facilitate the integration of existing applications into the LiSA Standard
- Mounting on these interfaces, the Developer Pack offers the dialog boxes

`LSADlgSelectSeeAlsoTuple`, `LSADlgEditHierarchyTuple`, `LSADlgSelectSeeAlsoTupleGlossary` and LiSA Script libraries that to a large extent manage the keyword administration.

- LiSA Configure is available for parameter administration

This minimal variant of the LiSA interface does not provide synonym administration; nor does it permit navigation through sparsely populated keyword hierarchies, ie. hierarchies in which although terms exist as "fathers" in the hierarchy of another keyword, they are no longer keywords in their own right.

The views are named with the aliases specified above. It is advisable to group the views with external names, so that the following picture results:

```
96. LSA-Minimal
      1. LSALuByHierarchy
      2. LSALuByLevel
      3. LSALuLog
      4. LSALuByIndex
      5. LSAPlByHierarchy
      6. LSAPlAlphabetic
```

Hierarchic structure in LiSA Glossary

There are three basic delimiters:

- The pipe ("|") as delimiter for different hierarchic levels
- The tilde ("~") as delimiter for various tuples (words, units) within any one hierarchic level
- The backslash ("\") as delimiter for simple, unilingual display of keywords in the index.

Accordingly, the full hierarchic display of a keyword triplet with, for instance, two hierarchic levels would be as follows:
| ~ | Tuple1_1 ~ Tuple1_2 ~ Tuple1_3 | Tuple2_1 ~ Tuple2_2 ~ Tuple"_3

- Delimiters deployed are declared at the beginning: "|~"
- A pipe then follows to indicate the beginning of the first hierarchic level: "|"
- The elements of the triplet in the first hierarchic level then follow, all three separated by a tilde: "~"
- A further pipe designates the beginning of the second hierarchic level: "|"
- The triplet of the second hierarchic level then follows, all three elements separated by a tilde: "~"
- The same conventions apply for additional hierarchic levels.

In the LiSA Glossary, the first tuple in every hierarchy level is assigned a time stamp with the following format: "Year/Month/Day Hour:Minute:Second". Thus, every keyword and current update status can be identified. The second tuple then contains the keyword in the first language (= main language), the second tuple contains the keyword in the second language, and so forth.
Example:
|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt

Views

Column	Title	Example	Description
LSALuByHierarchy			
1	LSAHierarchyPath&LSAHierarchyNode	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~").
2	DocumentUniqueID	3FBBC444F484C8EFC125669D0059ECF0	Unique identification of the document.
3	LSAHierarchyViewIndex	Country\Federal state\Capital	Simple, uni-language representation of keywork, e.g. for usage in categorized views. Backslash "\" functions as delimiter.
LSALuByLevel			
1	LSAHierarchyLevels	2|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland	Hierarchic level of keyword ("2") and full hierarchic path of keyword. This is the by which the keywords available for selection on a specific level in a specific branch can be located.
2	LSAHierarchyNode	1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of the keyword.
LSALuLog			
1	LSALogOldKeyword	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of the old keyword. This coumn is used as key to update keywords used in other documents. If a keyword corresponding to this protocol entry is located, it is replaced by the appropriate new keyword in the second column.
2	LSALogNewKeyword	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/10/14 18:44:13~Capital city~Hauptstadt	Full, hierarchic representation of the new keyword. ("Capital" was changed to "Capital city".)
LSALuByIndex			
1	LSAHierarchyViewIndex	Country\Federal state\Capital	Simple, uni-language representation of keywork, e.g. for usage in categorized views. Backslash "\" functions as delimiter.
2	DocUniqueID	3FBBC444F484C8EFC125669D0059ECF0	Unique document identification.
3	LSAHierarchyPath&LSAHierarchyNode	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~").
LSAPLByHierarchy			
1	Internal key (concealed)	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~").
2	Arbitrary hierarchic display	**Country** **Federal state**	This view is conceived for usage in pick lists. The result returned should be the unique key of the first column. The hierarchic display is freely definable.
3	Arbitrary identifier of lowest level	Capital	The element name is freely definable likewise.
4	Arbitrary	arbitrary	
LSALuAlphabetic			
1	Internal key (concealed)	|~|1998/07/15 09:58:56~Country~Land|1998/07/03	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"),

		10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	tuples by tilde ("~").
2	Arbitrary identifier of lowest level	Capital	This view is conceived for usage in pick lists. The result returned should be the unique key of the first column. The hierarchic display is freely definable. The element name displayed is freely definable. The column is sorted.
3	arbitrary	arbitrary	



Document: 2. Full

The Full interface of the LiSA Glossary permits the usage of synonyms and navigation through sparsely populated keyword hierarchies, i.e. ones in which terms exist only as "fathers" of a different keyword, but are no longer keywords in their own right.

The Full interface definition of LiSA Glossary is limited, like the Minimal version, to the standardization of Lotus Notes views. The following views are required as standard:

1. *LSALuBySynonym* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. This view contains only those valid keywords for which a synonym has been defined. If a synonym is chosen when a keyword is selected, then this view functions as the link that connects the keyword path and synonym (1st column) with the keyword path and keyword (4th column). This means synonyms can be replaced by the actual keywords.
2. *LSALuByByLevelSparseSynonym* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. An advanced version of LSALuByLevel, this view is used to navigate through "sparsely populated hierarchies" that lack certain nodes. In the first column, every keyword is displayed with its synonym on every hierarchic level; i.e. a keyword "State" with the keyword path "Earth\Continent\" is displayed both on the zero and first ("Earth") and second ("Continent") hierarchic levels. This renders superfluous the need for a separate entry for every keyword, and the redundant keyword paths make it possible to locate keyword entries even if no keyword actually exists at a given point. The second column shows the entire keyword path together with either the corresponding keyword or the appropriate synonym. Keywords and synonyms are shown.
3. *LSALuByLevel0SparseSynonym* is a view for the `@DBLookups` and the `GetDocumentByKey` method of the Notes class `Notesview`. At zero level in a hierarchy it is relatively likely that a `@DbLookup` command will be expected to return more than 64K, a volume that would result in an error message and termination. For this reason, the view supplements the *LSALuByByLevelSparseSynonym* view described above and is a way of overcoming the 64-K limit applying to `@DbLookups` commands. The view consists of only one column showing as separate entries all keywords and synonyms, but only on the zero level, and categorizing them as such. In consequence, merely the keywords on level zero remain in existence. This small number of entries can then be read-out effortlessly with the `@DBColumn` command

No standardization of fields has been undertaken, due primarily to the following two considerations.

- It is desirable to keep the minimal interface as narrow as possible in order to facilitate the integration of existing applications into the LiSA Standard
- Mounting on these interfaces, the Developer Pack offers the dialog boxes `LSADlgSelectSeeAlsoTuple`, `LSADlgEditHierarchyTuple`, `LSADlgSelectSeeAlsoTupleGlossary` and LiSA Script libraries that to a large extent manage the keyword administration.
- LiSA Configure is available for parameter administration

The views are named with the aliases specified above. It is advisable to group the views with external names, so that the following picture results:

```
97. LSA-Full
      1. LSALuBySynonym
      2. LSALuByLevelSparseSynonym
      3. LSALuByLevel0SparseSynonym
```

Hierarchic structure in LiSA Glossary

See Minimal interface definition of LiSA Glossary.

Views

Column	Title	Example	Description
LSALuBySynonym			
1	LSAHierarchyPath&Synonym	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|0000/00/00 00:00:00~Capital town~Regierungssitz	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~"). The notation "0000/00/00 00:00:00" implies a synonym.
2	DocumentUniqueID	3FBBC444F484C8EFC125669D0059ECF0	Unique identification of a document.
3	LSAHierarchyViewIndex	Country\Federal state\Capital	Simple, uni-language representation of keywork, e.g. for usage in categorized views. Backslash "\" functions as delimiter.
4	LSAHierarchyPath&LSAHierarchyNode	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt	Full, hierarchic representation of a keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~").
LSALuByLevelSparseSynonym			
1	LSAHierarchyLevels	**0**|~| **1**|~|1998/07/15 09:58:56~Country~Land **2**|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland	The keyword reads: |~|1998/07/15 09:58:56~Country~Land |1998/07/03 10:57:14~Federal state~Bundesland |1998/09/24 15:24:06~Capital~Hauptstadt and is thus shown **three times**: at Level 0, Level 1, and Level 2
2	LSAHierarchyPath&LSAHierarchyNode or Synonym	|~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|1998/09/24 15:24:06~Capital~Hauptstadt**,** |~|1998/07/15 09:58:56~Country~Land|1998/07/03 10:57:14~Federal state~Bundesland|0000/00/00 00:00:00~Capital town~Regierungssitz	Full, hierarchic representation of the keyword. Hierarchic levels are delimited by pipe ("|"), tuples by tilde ("~"). The notation "0000/00/00 00:00:00" implies a synonym.The keyword and synonym are delimited by commas. These entries are returned as a list by the Lookup command.
LSALuByLevel0SparseSynonym			
1	LSAHierarchyPath&LSAHierarchyNode or Synonym	1998/07/15 09:58:56~Country~Land	Although this view contains all keyword documents, it only shows the first level of the keyword path, categorizes these entries, and folds together the categories.

Chapter: LiSA-Help

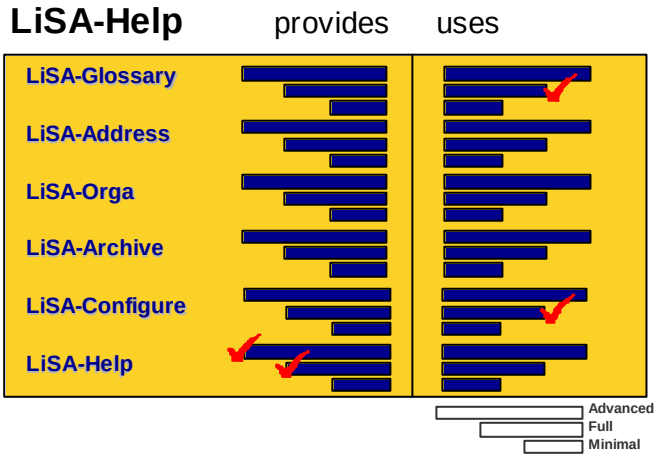
LiSA Help provides interfaces only at the Full and Advanced levels. Since the internal structures of this module are so complex that an alternative implementation is unlikely to be considered, no details of its implementation in the scope of the Minimal interface are given.

Instead, LiSA Help provides design blocks making possible simple and rapid integration with the context-sensitive help function in any desired Notes application.

The view for context-related access is defined as the Full interface. This allows a customized implementation of a direct help access.

The description of the type structure is added with the Advanced interface. This makes it possible to select LiSA Help documents from a database context. However, specific design elements and agents must be present in the database from which help is being requested. Subject to certain restrictions (help via Smarticon only, no field help), however, a help callup is also possible without modification of the application. "Provided design blocks" in this document offers a brief description of this integration.

LiSA Help uses the interfaces to LiSA Glossary and LiSA Configure.





Document: 1. Ready to use design blocks

The following design elements are provided to implement a LiSA conformal, consistent, context sensitive Help.

Script Libraries

Title	Description
LiSA-Library Help	
~~LSATxtHelp~~	~~This Script library contains all language specific messages used in the agent "(LSAHelpContext)"~~ *This lib is discontinued due to the improved handling of error messages with the LiSA Messagehandler*
LSALibHelpAgt	**All common code has been seperated out of the agents into this script library which also includes the classes "LSALibHelpMgr" and "Context"**
LiSA-Library Messagehandler	
LSALibMsgHHelpAgt	**This new library contains the class definition for the agent's error messages and is generated by the messagehandler**
LSALibMsgH	**This new library provides the base class for error message handling and is generated by the messagehandler**
LSALibMsgHUtil	**This new library is required as it makes some utility functions available for the messagehandler**
LiSA-Library Basic	
LSALibFct	**LSALibMsgHUtil**

Agents

Title	Description
LiSA-Library Help	
(LSAHelpF1)	This agent is necessary in every application wishing to call up help via F!
(LSAHelpContext)	This agent is necessary in all applications requiring direct help with pre-defined key, Vvia F1 or Smarticon. **This Agent has been redesigned towards the functions in LSALibHelpAgt**
(LSAHelpContextWEB)	**This new Agent uses the functions in LSALibHelpAgt and provides the context sensitive help for the WEB**

Integration of SmartIcon help

The following steps are necessary in order to activate the Smarticon help for an application:

1. Define your help document, and while doing so stipulate a unique document identifier "(Alias)" . If no such field appears in the help document, then ask your help administrator to alter the document type. You may also specify a custom-defined additional context. The alias used here must correspond to the design element.

Application	any name possible
Template "About the database"	alias of database (see also LiSA Configure)
View	alias of view
Form	alias of the Form
Field	field name

2. The context can be checked in a view in the capacity of author. As the illustration shows, internal identifier and type identifier alternate in the form of a hierarchy. This column therefore shows all possible keys in the way they must be used in the callup.

```
LSADEVPACK~TEMPLATEGROUP|LSAUDFHELP~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLADDR~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLBASIC~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLGLOS~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT|*~USING
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT|*~USING|*~SCENARIO
```

The view LSALuHierarchyForAccess is used during access as entry into the program.

3. For all databases for which help is to be possible, the agents and the script libraries above must be inserted from the LiSA Library Help. We recommend that this be done for all databases.

4. The Smarticon  for context-sensitive help is added to the desktop with the following code. To do so, select the Smarticon option offered in the menu choice "File - Tools - Smarticons", move the Smarticon to the right-hand side with "drag and drop", and edit it with "Edit Icon" in order to assign it a name, e.g. "Application Help".

After installation, the necessary files are located in the corresponding directory (notes\data\w32):

- lsahelp.bmp
- lsahelp.mac.

The following code of the Smarticon is contained in the file "LSAhelp.mac" .

```
_TXT_OnWorkspace := "Click <F1> to get context-sensitive help on a database selected
                    You cannot use the smarticon to call context-sensitive help.";
REM "Find the physical database name of the current database";

_DBServer := @Name([Abbreviate] ; @Left(@Implode(@UpperCase(@DbName); "!!"); "!!"));
_DBName := @Right(@Implode(@UpperCase(@DbName); "!!"); "!!");
_CurrentDB := _DBServer + "!!" + _DBName;

@If(@Trim(_CurrentDB) = "!!"; @Return(@Prompt([OK]; _TXT_TitleHelp; _TXT_OnWorkspace
@Command([ToolsRunMacro]; "(LSAHelpF1)")
```

The code has been simplified dramaticaly. The draw back of that is the NEED of the agents in the target database!

Integration of F1 help

If you additionally want to activate F1 help, carry out the following steps supplementary to those for Smarticon:

1. Then provide the standard Notes help with the appropriate instrumentation:

- Insert into the standard Notes help the view LSA-Help\Context Help from the LiSA Library Help.
- This view now shows the documents having to be instrumentalized. Some 6 in number, these documents are the ones displayed when F1 is pressed. Each of these documents is now provided with the hotspot shown below. A current version is included in the document"Integration help into your environment/database" in the LiSA Library Help.

  Context sensitive help

- The number of documents proved to be duplicated in some cases; this can be remedied by deleting those documents not provided with the new hotspot.

Integration of direct help with pre-defined context

If you wish to **directly** access LiSA Help per program, this can be accomplished with a **pre-defined key** describing the context. Possible definitions include, for example,

an action 

or button 

used to pre-select specific help pages. This option is implemented in LiSA Help, and requires the following steps:

1. Define your help document, and while doing so stipulate an unambiguous "(Alias)" . If no such field appears in the help document, then ask your help administrator to alter the document type. You may also specify a self-defined context as alternative.

2. The view **LSALUHierarchyForAccess** now shows the context with which access to the document is possible**:**

```
LSADEVPACK~TEMPLATEGROUP|LSAUDFHELP~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLADDR~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLBASIC~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLGLOS~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT|*~USING
LSADEVPACK~TEMPLATEGROUP|LSLHELP~ABOUT|*~USING|*~SCENARIO
```

As the illustration shows, internal identifier and type identifier alternate in the form of a hierarchy. This column therefore shows all possible keys in the way they must be used in the callup. To obtain this key, open a help document as author, and press the `Copy to clipboard` button in the "Context" section. The corresponding context key can then be inserted in the code example (second line, line below 4).

3. For all databases for which help is to be possible, the agents and the script libraries above must be inserted from the LiSA Library Help. We recommend that this be done for all databases.

4. Provide the button or action with the following code:

```
REM "define the context string";
_context:="LSADEMO~TemplateGroup|LSDHELP~about|Demo~Form":
      "LSADEMO~TemplateGroup|LSDHELP~about|Demo~Form|Subject~Field";

REM "Set the parameter";
_udf := "UDF" : "NoCache";
_Value:=@DbCommand("udf":"NoCache";"@CodeParam";_Context);
@DbCommand("udf":"NoCache";"@SetParam";"LSAPredefinedKey";1;_value);

REM "Define if the code is called from the help-database (Yes) or not (No)";
@DbCommand(_udf; "@SetParam";"LSALaunchInHelpDB"; 1; "No");

REM "Call the help agent";
@Command([ToolsRunMacro]; "(LSAHelpContext)");
@Command([ToolsRunMacro];"(Close)")
```

Integration of context sensitiv help in the WEB

To provide a user with a context sensitive help it is necessary to find out as much information about the current context as possible. This task is performed by a help agent sitting in each database. The agent will loop through all frames and collect this information and search the corresponding help documents. In a Notes client information about the current view or form can be found via Lotus Script. The information is not available in a browser. Therefore each page and form (views are displayed via the form $$ViewTemplateDefault) must provide the agent with its own context information. This is reached by inserting a hidden Pass-Thru HTML sequence that defines a field readable by the agent. This field name must have the name "LSAHelpIntContext".

Include the following pass-thru HTML text

- **in a page:**
  `<input type="hidden" name="LSAHelpIntContext" value="computed text">`

- **in a form:**
  `<input type="hidden" name="LSAHelpIntContext" value="include a computed for display field here">`

The computed text and the field (also named LSAHelpIntContext) contain the following formula:

`"*~TemplateGroup|*~About|name of the design element~type of the design element"`

Example:
`"*~TemplateGroup|*~About|LSAContact~Form"`

☐ **Note: Make sure that each form contains this code with the name of the form hardcoded. Do not try to compute the form name via the form field, because this field is not available in new documents and documents where the form is stored with the the document.**

The Help Tab found on each LiSA Application provides the following HTML code in the LSAStdNavPage

```
<head>
<script language="javascript">
function getHelp()
{
var thisDB = 'lsa/EN/content/lsadpck.nsf';
callHelp(thisDB);
}
</script>
</head>
```

☐ **Note: The database name is computed text!!!**

This function is called by the help button via Java script
`onClick="getHelp()" Target="_self" HREF="LSAStdNavPage?openPage"`

☐ **Note: The Java Script CallHelp function is provided as JSHeader in the LSAStdNavPage it loops through the frames, collects the contexts and calls the help agent `LSAHelpContextWEB`. For all databases for which WEB help is to be , the agents and the script libraries above must be inserted from the LiSA Library Help. We recommend that this be done for all databases.**

Usage of the Full interface allows an application the options of both

- context-sensitive

LiSA Help, as well as

- direct callup via a pre-defined key.

If an application adheres to the Full interface, it can use the LiSA Help as its own help in a team or in an application. Although the LiSA Help template is used in this process, it is not necessary for the type structure to be used in the specified way, meaning either a problem-adapted or more simple structure can be chosen. In consequence, it is not possible to integrate such help documents into the standard LiSA Help. Nor can the context be determined automatically, since this requires a specific type tree attainable only with the pre-defined types of the Advanced interface.

As a component of the Full interface, an access view as shown below is provided by the LiSA Help.

```
97. LSA-Full
      1. LSALUHierarchyForAccess
```

Views

Column	Title	Example	Description
LSALuHierarchyForAccess			
1	LSAAllContextsForAccessCalc	z.B. |~|LSAKernel~TemplateGroup|LSAHelp~About |~|LSAKernel~TemplateGroup|LSAHelp~About|Type~Form |~|*~TemplateGroup|LSAHelp~About |~|*~TemplateGroup|LSAHelp~About|Type~Form	The context key is a hierarchy of pairs consisting of alias and type. Every alias is freely definable, while the type is derived from the corresponding structure in Help. Every context is displayed twice: once with the application, then a second time with * as application name.

Direct help callup

Direct help via pre-defined key is called up with a code piece able to be implemented as a hotspot, action, etc., at virtually any desired point in the design of an application. See the chapter "Provided design blocks" in this document for a description.

Document: 3. Advanced

The Advanced interface to the LiSA Help has two main advances over the standard help offered:

1. Context-sensitive access to data is possible due to definition of help structure in a type tree.
2. The uniform structure allows several databases to be combined in a shared database via copying/inserting.

   In order to guarantee these options, the standard types should never be modified.

Help callup with F1 and Smarticon

Context-sensitive help can be called up with the F1 key on the one hand, and with the Smarticon  on the other.

For callup via F1, a hotspot labeled "Context-sensitive help" was integrated in several standard Notes help documents.

Both the hotspot activated via F1 and the Smarticon activate one and the same code, which initiates context determination and opens the help database with the relevant help document. See "Provided design blocks" in this document for a detailed description.

Standardized help type structure

It is essential that the type structure shown below be used. The types labeled with * are used for context detection, meaning the internal type designations shown in bold print are mandatory. In order to be able to use help documents produced independently of each other , the type tree supplied must be used without prior modification!

Application)[1]	**TemplateGroup**)[2]	*
Template	**About**	*
Agent	Agent	
View **View**	*	
Button	Button	
View-Actions	ViewAction	
Dialogbox	Dialogbox	
Field	Field	
Button	Button	
Shared Field	SharedField	
Frequent Questions	FAQ	
Implementation	Implementation	
Scenario	Scenario	
...	...	(nested as deeply as required)
Chapter	Chapter	
...	...	
Scenario	Scenario	
...	...	
Form **Form**	*	
Field	**Field**	*
Button	Button	
Form-Action	FormAction	
Navigator	Navigator	
Script Library	ScriptLibrary	
Subform	Subform	
Field	Field	
Buton	Button	
How can I	**Using**	*
Chapter	Chapter	
...	...	
Scenario	Scenario	
...	...	
Scenario	Scenario	
...	...	
How can I	Using	
Chapter	Chapter	
...	...	
Scenario	Scenario	
...	...	
Scenario	Scenario	
...	...	

)[1] the left column is language dependent while

)[2] the right column represents the internal names. These must not be changed!

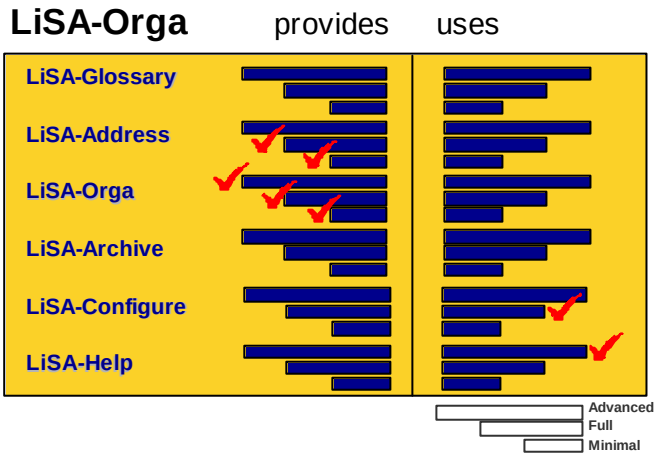
Chapter: LiSA-Organization

LiSA Orga offers interfaces at the Minimal, Full and Advanced levels and mounting on the interfaces of the core components LiSA Configure and LiSA Help. LiSA Orga contains address data and accordingly offers the LiSA Address interfaces documented in this Docu-Pack.

A number of views allowing organizational data to be read are defined at Minimal level. These include pick lists and views for access by key through programs.

At Full level, there are additional views relating especially to organization type and meta-modeling.

Defined at Advanced level are Script functions unable to be simply implemented by means of views.





Document: 1. Minimal

The Minimal interface definition of LiSA Organization is limited to the standardization of Lotus Notes views and pick lists.

An initial category of views has been optimized for pick list output, in line with the following distinction:

- persons and
- organizational units

are to be found in different views. Similarly, a distinction is made between views imaging the Notes name and address book and those included in the dabase purely for purposes of information.

The following views are assumed to be standard:

1. *LSAOrgPLPerson* is a view presented to the user in a window in response to the command `@Picklist`. It contains all person-related documents. Once the user selects a document, as a result the first column (concealed, unsorted) should be returned containing an ID as unique internal key able to be used for reading-out the serial data of the selected document in the view *LSAOrgLuAllByID.* The other columns serve the purpose of preparing the information for the user in clear form. The column buttons can contain Viewlinks leading to other sortings of the same document, e.g. according to surname, or forename, while the top view is sorted according to a suitably composed visible display text.
2. *LSAOrgPLPersonNAB* is a view which is analogous with *LSLOrgPLPerson*. However, it contains only person-related documents that include NAB-relevant entries (basically all active Notes users).
3. *LSAOrgPLUnit* is a view presented to the user in a window in response to the command `@Picklist`. It contains all documents describing organizational structures. Once the user selects a document, as a result the first column (concealed, unsorted) should be returned containing an ID as unique internal key able to be used for reading-out the serial data of the selected document in the view *LSAOrgLuAllByID.* The other columns serve the purpose of preparing the information for the user in clear form. The column buttons can contain Viewlinks leading to other sortings of the same document, e.g. hierarchically or by type, while the top view is sorted according to a suitably composed visible display text.
4. *LSAOrgPLUnitNAB* is a view which is analogous with *LSLOrgPLUnit*. It contains all documents that describe organizational structures and include NAB-relevant entries (basically all organizational units managed in the NAB for the access control groups).
5. *LSAOrgLUAllByID* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All type-, person-, and organizional-unit-related documents **and Traits** appear in this view. The first column is sorted and contains an unambiguous internal key for accessing the information in the documents. The second column contains the `@DocumentUniqueId`, and may therefore also be used for re-coding, e.g. in order to parametrize an `@Command([OpenDocument])`. The third column is a results column and contains the serial document data. **A fourth and fifth column has been appended to allow quick decoding of internal IDs into an external representation like the default hierarchy or an unique organizational key. This remaines empty for traits.**
6. *LSAOrgLUAllByDocUNID* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All type-, person-, and organizional-unit-related documents appear in this view. The first column is sorted and contains an unambiguous internal key for accessing the information in the documents. The second column contains the `@DocumentUniqueId`, and may therefore also be used for re-coding in order to access additonal data of the Full interface. The third column is a results column and contains the serial document data. It must be stressed once again that it is imperative to avoid field access in the

sense of the interface. Even if access is theoretically possible via the Document UNID, the view route must be taken.

7. *LSAOrgLUAllByNABName* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All type-, person-, and organizional-unit-related documents reflected in the NAB appear in this view (i.e. group-generating units or attributes, and also people). The first column is sorted and contains the Notes name, e.g. `@username` or alternatively one of the specified group names for accessing the information in the documents. The second column contains the `@DocumentUniqueId`, and may therefore also be used for re-coding in order to access additonal data of the Full interface. The third column is a results column and contains the serial document data.
8. *LSAOrgLUTraitsByID* is now part of the Full interface

This data access is deliberately not handled via fields, meaning no field standardizations has been undertaken. This is due primarily to the following two considerations.

- It is desirable to keep the minimal interface as narrow as possible in order to facilitate the integration of existing applications into the LiSA Standard.
- Due to the combined calculation of data in one interface field of a document, replication and storage conflicts almost inevitably arise when changes are simultaneously made to one document by two people or when changes are made from a decentral location. These conflicts cannot be automatically solved by the "Merge" option offered by Notes.

In order to avoid restricting the capabilities of Lotus Notes in this regard, the existence of merely a small number of views can be guaranteed at minimal level. This limited number of fields unfortunately means that the full bandwidth of the search functionality available in Notes is not available.

The views are named with the aliases stated above. We recommend that the views be grouped with an external name, so that the following picture results:

```
96. LSA-Minimal
      1. LSAOrgPLPerson
      2. LSAOrgPLPersonNAB
      2. LSAOrgPLUnit
      4. LSAOrgPLUnitNAB
      5. LSAOrgLUAllByID
      6. LSAOrgLUAllByDocUNID
      7. LSAOrgLUAllByNABName
```

Views

Column	Title	Example	Description
LSAOrgPLPerson or LSAOrgPLPersonNAB			
1	Internal key (unsorted, concealed)	0279Xf386	Internal key for data access (calculated ID in example)
2	Name1	Cybertech\Sales	This view is conceived for usage in pick lists. The result returned should be the unique key of the first column. The displayed name of a person is freely definable.
LSAOrgPLUnit bzw. LSAOrgPLUnitNAB			
1	Internal key (unsorted, concealed)	0279Xf386	Internal key for data access (calculated ID in example)
2	Name	Beliebig, abernicht leer	This view is conceived for usage in pick lists. The result returned should be the unique key of the first column. The displayed name of a person is freely definable.
LSAOrgLUAllByID			
1	Internal key (sorted)	0279Xf386	Internal key for data access (calculated ID in example)
2	DocumentUniqueID	B582586B9AD3B69EC1256636005EDF3C	Internal key for data access (DocumentUniqueID in example)
3	Defaultdata	|#~|LotusU#CCNotes#G#Departement#5EDF4443C#ACLGrDev~ACLGrDoku# #AB123457~XY123457# #6FDF446FD~3456FDF44~32435CF44 ###**CC23#Notes Application Development** |Client#free data ...	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" ,"LotusL" and customer data. See description "Data format of the business objects", below, for description.
4	**DefaultHierarchy**	**ACME\Sales\Central\Support**	**This is the default hierarchy, which is the path from one selected root unit down to the Unit.**
5	**Organizational Key**	**CC23**	**this is a abbreviation for the Unit that needs to be "unique" in the LSA-Orga. It may be generated automatically within the hierarchy**
LSAOrgLUAllByDocUNID			
1	DocumentUniqueID (sorted)	B582586B9AD3B69EC1256636005EDF3C	Internal key for data access (DocumentUniqueID in example)
2	Internal key	0279Xf386	
3	Defaultdata	|#~|LotusU#CCNotes#G#Departement#5EDF4443C#ACLGrDev~ACLGrDoku# #AB123457~XY123457# #6FDF446FD~3456FDF44~32435CF44 ###**CC23#Notes Application Development** |Client#free data ...	A serial text string containing the data blocks "LotusU", "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" ,"LotusL" and customer data. See description "Data format of the business objects", below, for description. **For traits the "LotusT" structure will be displayed**

LSAOrgLUAllByNABName			
1	NABName (sorted)	CN=Martin Muster/OU=Sales/O=ACME	Name as stored in NAB as key for data access (Username in example)
2	Internal key	0279Xf386	Internal key for data access (calculated ID in example)
3	Defaultdata	|#~|LotusU#DevNotes#G#Departement#5EDF4443C#ACLGrDev~ACLGrDoku##AB123457~XY123457##6FDF446FD~3456FDF44~32435CF44###**CC23#Notes Application Development**|Client#free data**...**	A serial text string containing the data blocks "LotusO", "LotusP", "LotusE", "LotusW", "LotusA" ,"LotusL" and customer data. See description "Data format of the business objects", below, for description.

Data format of the business objects

Business object <LSA SerialDataName>	Data block position	Attributes
LiSA Organization		
LotusU	<LotusU>.0	DATA BLOCK DESCRIPTION:Organization-specific data
	<LotusU>.1	**Abbreviated name** of unit or person as displayed
	<LotusU>.2	Class ID P=Person type, G=Group, C=TraitI
	<LotusU>.3	Name of associated type in display
	<LotusU>.4	Internal type key (for future extensions)
	<LotusU>.5.0	First NAB group name
	<LotusU>.5.1	Second NAB group name
	<LotusU>.5....	Additional NAB group names
	<LotusU>.6.0	nolonger used for parent Displaynames (see column 4 of LsaOrgLUAllByID)
	<LotusU>.7.0	Internal key of first father unit
	<LotusU>.7.1	Internal key of second father unit
	<LotusU>.7....	Internal key of additional father units
	<LotusU>.8.0	nolonger used for Child display names (see column 5 of LsaOrgLUAllByID)
	<LotusU>.9.0	Internal key of first member unit
	<LotusU>.9.1	Internal key of second member unit
	<LotusU>.9....	Internal key of additional member units
	<LotusU>10.0	nolonger used for TraitDisplayName (see LsaOrgLUTraitsByID)
	<LotusU>11.0	Inolonger used for TraitID (see column LsaOrgLUTraitsByID)
	<LotusU>.12	**Organizational key (planned future extensions)**
	<LotusU>.13	**Long name (planned future extensions)**
Client Data	<...>.0	DATA BLOCK DESCRIPTION:Customer data freely defined as serial text data
	...	

- **Hint: Due to limitations in Lotus Notes/Domino (15 KB for each field shown in a view, 63 KB for all fields shown in a view the structure <LotusU> was likely to get too big in larger organizations. Depending on the length of the names, the limit was at 100-200 entries for Parents, members and traits. As a consequence, this version does not keep the trait and hierarchical names in the structure <LotusU> anymore. The names are obtained via additional columns in the views**

  `5. LSAOrgLUAllByID`

  an additional view is provided for the traits in the full interface

- **Hint: Due to the same limitations we need to cut down the length of the hierarchical strings. Practical experience shows that an organization usually has a**
  1. **long name <LotusU>.13**
     like "Notes Application Development"
  2. **abbreviated name <LotusU>.1**
     like"CCNotes" that may not necessarily be unique on its own but becomes unique in the Hierarchy of a support organization like "ACME\Sales\Central\Support\CCNotes".
  3. **organizational key <LotusU>.12 and column 5 in LSAOrgLUAllByID**
     like "CC23" which is unique. For migration we will use the currently unique name as organizational key.

The type can be hardwired in Minimal standard, meaning a group might be titled "ProjectTeam", "Department", "Crew" or similar; the internal key of the unit would then have no entry.



Document: 2. Full

Alongside the standardization of further Lotus Notes views and pick lists, the Full interface of LSA Organization contains in particular ones for Traits as well as the provision of specific fields (with standardized names), functions, and options.

The following views are required as standard:

1. *LSAOrgPLTrait* is a view presented to the user in a window in response to the command `@Picklist`. It contains all trait-related documents. Once the user selects a document, as a result the first column (concealed, unsorted) should be returned containing an ID as unique internal key able to be used for reading-out the serial data of the selected document in the view *LSAOrgLuAllByID.* The other columns serve the purpose of preparing the information for the user in clear form. The column buttons can contain Viewlinks leading to other sortings of the same document, e.g. according to type, while the top view is sorted according to a suitably composed visible display text.
2. *LSAOrgPLTraitNAB* is a view which is analogous with *LSLOrgPLTrait*. It contains all trait documents that include NAB-relevant entries (basically the groups managed in the NAB).
3. *LSAOrgLUTypes* is a view allowing `@DbColumn` to be used to read all possible types from the first column or else to re-code them to the internal code in the second column. This option is implemented for subsequent enhancements.
4. ***LSAOrgLUTraitsByName* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All trait documents appear in this view. It is used as a mapping view between the external trait name and its internal key in the second column. In the third column you can retrieve the full trait information.**
5. ***LSAOrgLUTraitsAllByTrait* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All person-, and organizional-unit-related documents that have trait appear in this view (i.e. group-generating units or attributes, and also people) categorized by the trait key. Given the trait key you can programatically find out, which unit has a given trait.**
6. ***LSAOrgLUTraitsByID* is conceived as a view for the command `@DBLookup` and the `GetDocumentByKey` methods of the Notes class `Notesview`. All person-, and organizional-unit-related documents appear in this view (i.e. group-generating units or attributes, and also people). Given a Unit's ID, the view allows to determine easily which capabilities the unit possesses by easily lookong up it's assigned traits. The serial data provies all information necessary for further utilizing the traits in the application's context. The first column is sorted and contains an unambiguous internal key uf the unit for access. The second column contains the `@DocumentUniqueId`, and may therefore also be used for re-coding in order to access additonal data of the Full interface.**

   - **Hint: The view LSAOrgLUAllByID has been split up, to allow larger numbers of relations. Currently the number is limited by the 15 KB capacity of the field. given a 100 byte hierarchical name this is a limit of 130 references to Parents, children or traits. By clearing up this space and the 10 byte key only, we are likely to reach the limit at a sum of 1400 parents, children and traits at the same time. see *LSAOrgLUTraitsByID* . The aditional columns are required, in order to stay below the 64 KB loockup limit.**
   - **Hint: A mapping between the trait's displayname and it's id is given in *LSAOrgLUTraitsByName* in the Full interface.**

In order to use the full search capabilities of Lotus Notes, the Full interface requires specific fields. With them, the full bandwidth of the search functionality available in Notes is not available. If in a full-text indexed database the option exists of obtaining documents by search command, a

field-name-dependent search method can therefore be used. However, it should be remembered that due to the combined calculation of data in one interface field of a document, replication and storage conflicts almost inevitably arise when changes are simultaneously made to one document by two people or when changes are made from a decentral location. These conflicts cannot be automatically solved by the "Merge" option offered by Notes, but the solution is acceptable for these search fields.

The interface definition defines the standardization of three fields for an extended search interface:

HINT
Due to limitations in Lotus Notes/Domino (15 KB for each field shown in a view) the searchfield for the unit was likely to get flood by lengthly hierarchical names in larger organizations. Depending on the lenght of the names, the limit was at 100-200 entries. As a consequence, future versions will not keep these names in the field *LSAOrgIntUnitSearch* anymore. Instead a field *LSAHierarchyViewCategory* which may be used to display the categories in views, will be used. The *LSAOrgIntUnitSearch* will contain the default hierarchical name of the unit and unit's free data only. This change will be implemented in the next version. In the extreme 15 KB of possible 63 KB can be gained.

1. *LSAOrgIntUnitSearch* is used to locate organizational unit documents. The standard prescribes that the default hierarchic name the organizational unit associated with the Notes document primarily must be entered in this field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
2. *LSAOrgIntPersonSearch* is used to locate person-specific documents. The standard prescribes that the name of the person must be filed in the field in the following form: "<Surname> <Forename>" and "<Forename> <Surname>". In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
3. *LSAOrgIntTraitSearch* is used to locate trait-specific documents. The standard prescribes that the name of the Trait must be filed in the field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
4. ***LSAHierarchyViewCategory* contains the hierarchy of all parents of the document as they are displayed in a category It is used in conjunction with the search. For units you will evaluate, both *LSAHierarchyViewCategory* and the field *LSAOrgIntUnitSearch.* For Persons you need to evaluate both *LSAHierarchyViewCategory* and the field *LSAOrgIntPersonSearch.*For traits it is sthe fields *LSAHierarchyViewCategory* and the *LSAOrgIntTraitSearch* accordingly.**

The fields only occur in connection with the applicable document classes, and can therefore be used for document selection in selection formulas.

The views are named with the aliases stated above. We recommend that the views be grouped with an external name, so that the following picture results:

```
97. LSA-Full
      1. LSAOrgPLTrait
      2. LSAOrgPLTraitNAB
      3. LSAOrgLUTypes
      4. LSAOrgLUTraitsByName
      5. LSAOrgLUAllByTraits
      6. LSAOrgLUTraitsByID
```

Fields

Structure	Example	Description
LSA-Organization.LSAOrgIntUnitSearch (Text, multi value)		
"<Name of organizational unit 1>": "<Name of organizational unit 2>": ... "<Name of organizational unit n>": "<arbitrary string 1>": "<arbitrary string 2>": ... "<arbitrary string m>"	"Cybertech\Sales"	This field is used to locate organizational unit documents. The standard prescribes that the hierarchic names (delimited by backslash) of those organizational units associated with the Notes document must be entered in this field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
LSA-Organization.LSAOrgIntPersonSearch (Text, multi value)		
"<Surname> <Forename>": "<Forename> <Surname>": "<arbitrary string 1>": "<arbitrary string 2>": ... "<arbitrary string m>"	"Groß Franz": "Franz Groß": "Franz P Groß, Kundenr. 4771"	This field is used to locate person-specific documents. The standard prescribes that the hierarchic names of the person must be filed in the field in the formats *Surname Forename* and *Forename Surname*. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts as search criteria in this field.
LSA-Organization.LSAOrgIntTraitSearch (Text, multi value)		
"<Name of trait 1>": ... "<arbitrary string m>"	"Verteiler Projektmanager" "PMMail"	This field is used to locate trait-related documents. The standard prescribes that the externally visible name of the *trait* be filed in the field. In dependence on the implementation, it is additionally possible to anchor a series of freely definable texts, e.g. the group name, as search criteria in this field.
LSA-Organization.LSAHierarchyViewCategory (Text, multi value)		
"<Name of organizational unit1>": **...** **"<Name of organizational unit n>"**	**"Cybertech\Sales\Notrh"**	**This field is used to locate documents, that have a hierarchical structure.The standard prescribes that the hierarchic names (delimited by backslash) of those organizational units being fathers of the Notes document must be entered in this field.**

Views

Column	Title	Example	Description
LSAOrgPLTrait, or LSAOrgPLTraitNAB			
1	Internal key (unsorted, hidden)	0279Xf386	Internal key for data access (calculated ID in example)
2	Name	Help author	This view is conceived for usage in pick lists. The result returned should be the unique key of the first column. The displayed name of the traits is freely definable.
LSAOrgLUTypes			
1	Display Name	Project Group	the name as known by the user
2	Internal key	0279Xf386	Internal key for data access (calculated ID in example)
LSAOrgLUTraitsByName			
1	**Display Name**	**Skill**	**the name as known by the user**
2	**Internal key**	**DE123457**	**Internal key for data access**
3	**Trait reference**	**|#~|LotusT#Skill#1# PartitionedCompetence# DE123333#grpSkill_#@left(Cl;1)**	
LSAOrgLUAllByTrait			
1	**Internal trait key (sorted)**	**DE123457**	**Internal key for trait access (calculated ID in example)**
2	**internal unit key**	**B582586B9AD3B69EC12566 36005EDF3C**	**Internal key for data access (DocumentUniqueID in example)**
LSAOrgLUTraitsByID			
1	**Internal key (sorted)**	**0279Xf386**	**Internal key for data access (calculated ID in example)**
2	**Internal trait key (sorted)**	**DE123457**	**Internal key for trait access (calculated ID in example)**

Data format of the business objects

Business object <LSA SerialDataName>	Data block position	Attributes
LiSA-Organization		
LotusM	<LotusM>.0	DATA BLOCK DESCRIPTION: reserved for later extension This will become the type of the Meta model
LotusT	**<LotusT>.0**	**DATA BLOCK DESCRIPTION: Trait info reserved for later extension**
	<LotusT>.1	**Display name of trait**
	<LotusT>.2	**SubClass Id of trait** **1 = standard static trait** **2 = formula based dynamic trait** **3 = relational trait**
	<LotusT>.3	**Name of associated type in display**
	<LotusT>.4	**Internal type key (for future extensions)**
	<LotusT>.5	**NAB group name or the fixed part of it**
	<LotusT>.6	**The partitioning formula (only for SubClass 2) (See Hint)**

- **Hint: the groupname and the formula may be empty in case of a trait that does not create groups. In case of partitioned traits the groupname only reflects the prefix while the rest is determined by a formula. If a program wants to use the groups it needs this formula for evaluation.**

In the Full standard, the type is read flexibly from a meta-model, meaning a group might have the name "ProjectTeam", "Department", "Crew" or similar, with the internal key referring to the type document.

With the ongoning improvement of the interface to organizaitional data, relations are introduced as advanced interface. Usualy the unit that holds a relational trait has information as follows

- **internal key of relational trait document**
- **relation symbolic name (e.g. "Manager")**
- **relation display name (e.g. "manages")**
- **reverse relation display name (e.g. "reportes to")**
- **private / public**
- **list of references to entities (internal keys approx 10 byte)**
- **list of entities display names (approx 30 byte)**

Hint:
The above structure is taken from the LSA-Orga implementation and serves as an example for relational traits. Other structures are possible. Only the view interface below is part of the standard.

Hint:
Given the field length of 15 KB and assuming that each relation entry is stored in a serialized format in a text list we can estimate a size of (110 + n *(10+30)) byte per relation which would be enough for

48 relations with 5 entities
16 relations with 20 entities
3 relations with 100 entities

Care has to be taken not to overflow these limits. Keep also in mind that the limit of summary data is 63 KB.

Reverse relation information is usually stored in the units that the relation refers to. The reverse information is like

- **internal key of relational trait document**
- **relation symbolic name (e.g. "Manager")**
- **relation display name (e.g. "manages")**
- **reverse relation display name (e.g. "reportes to")**
- **private / public**
- **reference to entity (internal keys approx 10 byte)**
- **display name of entity (approx 30 byte)**

Hint:
The above structure is taken from the LSA-Orga implementation and serves as an example for relational traits. Other structures are possible. Only the view interface below is part of the standard.

Hint:
Given the field length of 15 KB and assuming that each reverse relation entry is stored in a serialized format in a text list we can estimate a size of (110 +(10+30)) byte per relation which would be enough for 100 reverse relations
Care has to be taken not to overflow these limits. Keep also in mind that the limit of summary data is 63 KB.

Views for programmatic access

Column	Title	Example	Description
LSAOrgLURelation			
1	Internal key~Relation Name (sorted)	012345678~Manager	Internal key for data access (calculated ID in example) + "~" + relational trait name
2	Internal key (s)	0279Xf386; 0279Xf387	List of internal keys of entities
LSAOrgLURevRelation			
1	Internal key~Relation Name (sorted)	0279Xf386~Manager	Internal key for data access (calculated ID in example) + "~" + relational trait name
2	Internal key	012345678	Internal key of entity



Chapter: LiSA-UDF

The LiSA Library UDF is a functional collection providing a series of functions both for@command language and for LotusScript. These functions enhance the functionality of Lotus Notes, e.g. @SetClipboard or @Message, and also include functions for version checking and access to LiSA Configure.

The capsulation of functionalities in one UDF function enables the central provision of a program logic which would otherwise have to be included at many points in applications. Furthermore, this method is the only way of ensuring acceptable response times for functions which are often complex. The function is called up over a uniform interface described in the example below:

Callup examples

@command language

```
_path := @DbCommand("UDF":"NoCache"; "@PathByAlias"; "LSAGLOS");
```

LotusScript

```
Dim u As UDF
Set u = New UDF(db.Server,db.FilePath)
ret   = u.PathByAlias("LSAGLOS")
```

The following table surveys the individual modules and functions of the LSA Library UDF. See the separate help for LSA Library UDF for a description of the function parameters and callup examples.

LSA Library UDF

UDF Main Module	
@CreateLSXEntry	Generates the registry entry for the LSX-DLL in the registry of the applicable platform
@ExeCall	Starts an application
@FindDesignNote	Checks whether a design element is included in a database
@GetModuleInfo	Fetches copyright and version information of all UDF modules
@GetOutput	Fetches the settings for the UDF output window
@GetOutputDest	Fetches the destination for UDF output
@GetVersion	Fetches the version number of the UDF Library and the available functions
@Message	Copies a text to the status line of Lotus Notes
@SetOutput	Sets the settings for the UDF output window
@SetOutputDest	Sets the destination of the UDF outputs (status line or UDF window)
@Trace	Writes Notes data to the UDF window
Configure (CFG)	
@DesignAliasByPath	Detects a database design alias in LSA Configure
@DbAliasByPath	Detects a database alias in LSA Configure
@CodeParam	Converts a Notes data type into a parameter (textual representation)
@DecodeParam	Converts a parameter (textual representation) into a Notes data type
@GetConfigurePath	Detects the current path to LSA Configure
@GetLocation	Detects the current LSA location
@GetParam	Fetches in a parameter (textual representation) for all ranges or a given range
@GetTeam	Detects current team
@Interface	Detects all interface names for a given database
@PathByAlias	Detects a database path in LSA Configure
@PathByInterface	Detects those databases supporting a specified interface
@SetParam	Sets value of a parameter in Notes.ini or temporarily
@VExpand	Substitutes parameter for dummy in format string
Miscellaneous (MISC)	
@Ascii	Supplies the ASCII code of the first character in a character string
@CompositeToText	Converts Rich Text to standard text
@DataType	Detects Notes data type on basis of value of a variable
@Error	Converts a number into corresponding error message, and vice-versa
@GetElements	Selects elements from a list on basis of position numbers
@NumberToHex	Converts a decimal nototation into hexadecimal (text)
@SetClipboard	Temporarily stores text
@Sort	Calculates sorting sequences for unsorted lists
@Sum	Calculates total of a list of numbers
@TextExpand	Searches character strings for dummies and replaces them
@Translate	Searches and replaces characters in a text or text list
@UserLogWrite	Copies a text to the log database of a user
Version Control (VCT)	
@VCTRead	Fetches version and copyright information from design elements
@VCTReport	Compiles a report in the VCT database
@VCTStamp	Stamps version and copyright information onto design elements
Archive	
Notes Freeze	See interface definitions LSA Archive Minimal and Full, as well as UDF Help
@LSAASetArcRules	Filing a document in a separate Notes database
@LSAASetArcRenderRules	
@LSAASetArcSource	

@LSAASetArcSourceField
@LSAASetArcTarget
@LSAASetArcTargetField
@LSAAArchive

Archive

@LSAAInit
@LSAAArcRetSetParam
@LSAAArcRetOpenConnection
@LSAAIdxRetSetParam
@LSAAIdxOpenConnection
@LSAASetArcRules
@LSAASetArcSource
@LSAASetArcSourceField
@LSAASetArcTarget
@LSAASetArcTargetField
@LSAAArchive
@LSAAIdxCloseConnection
@LSAAArcRetCloseConnection
@LSAATerminate

Archiving a Notes document with separate archive middleware (not included in scope of LSA contents)

Search

@LSAAInit
@LSAAArcRetSetParam
@LSAAArcRetOpenConnection
@LSAAIdxRetSetParam
@LSAAIdxOpenConnection
@LSAASetQuery
@LSAASetQueryRules
@LSAASetQueryTarget
@LSAASetQueryTargetField
@LSAAQuery
@LSAAIdxCloseConnection
@LSAAArcRetCloseConnection
@LSAATerminate

Search through index for a document

Retrieval

@LSAAInit
@LSAAArcRetSetParam
@LSAAArcRetOpenConnection
@LSAASetRetSource
@LSAASetRetTarget
@LSAARetrieve
@LSAAArcRetCloseConnection
@LSAATerminate

Relocation of archived information