Guide to version 2.1

August 8, 2011

www.qkom.net

Q!kom

Table of Contents:

Table of Contents: 1
Preface 2
Introduction 3
Installation and system requirements 4
What Is New 5
Licensing 6
The soapgate Q! database 8
- Navigation & Views 8
- Configuration & Maintenance 9
- Database Access Profiles (new) 12

Consuming soapgate Q! 19
API Reference 21
- dbACL 22
- dbCallAgent 23
- dbClearFolder (new) 24
- dbColumn 25
- dbColumnX (changed) 26
- dbDeleteDoc 27
- dbDocAttachmentList (new) 28
- dbDownloadFile (new) 29
- dbEffectiveRights 30
- dbFTSearch (changed) 31
- dbGetDelStubs (new) 32
- dbGetFieldTypes 33
- dbGetFormFields 34
- dbGetMailInfo 35
- dbLookup 36
- dbLookupX (changed) 37
- dbOutlineViews 39
- dbPutInFolder (new) 40
- dbReadDocFields 41
- dbReadProfileFields 43
- dbRemoveFromFolder (new) 44
- dbRenderDoc 45
- dbRowX (changed) 46
- dbSaveDocFields 47
- dbSaveProfileFields 48
- dbSearch (changed) 49
- dbSendDocument 50
- dbSessionKeepAlive 51
- dbUploadFile (new) 52
- dbUserRoles 53
- dbView2XML (changed) 54
- dbViewColumns 56
- dbViewFTSearch (changed) 57
- dbViews 58

Appendix 59
- Notes field and data types: 59
- Notes Database Access Levels: 60
- Data Serialization 60

Preface

We thank you for being a user of soapgate Q!, - the universal web service based Data Access API for Lotus Notes Domino™.

This manual is designed to assist Domino administrators in the installation of soapgate Q! as well as developers in the use of the web service API provided by soapgate Q!

Every reasonable attempt has been made to ensure the accuracy of this user manual and that it reflects the operations of the product, however users of soapgate Q! are responsible for ensuring that the product and its documentation are suitable for the needs of that organization. No warranties in respect of this user manual are being made or can be assumed.

Introduction

soapgate Q! is a universal data access API for Domino based on a web service. The web services are contained in a single web service stored in the soapgate Q! (Notes) database.

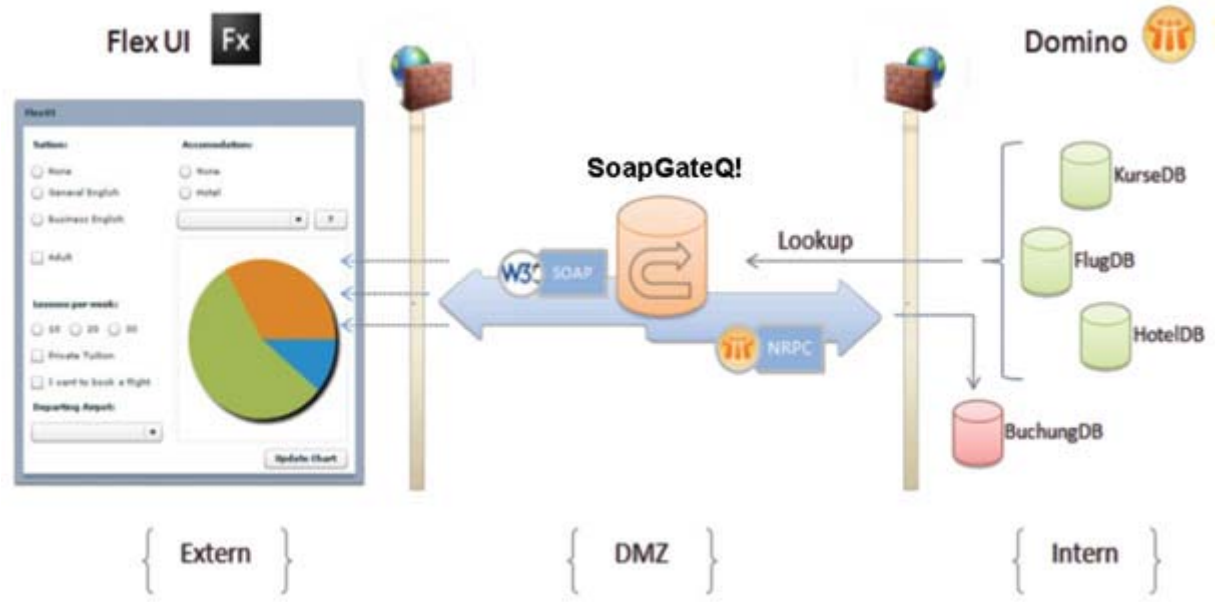
The API consists of a collection of native Notes Formula and Lotus Script classes (or methods respectively), such as @dbcolumn(), @dblookup(), NotesDatabase.DBSearch(), NotesView.FTSearch etc...

In addition to the native Notes features the API also contains a few web service operations that provide extended functionality, such as dbcolumnX() and dblookupX(), which unlike their native counterparts return more than one column or field in a single call.

The concept of soapgate Q! is to provide an easy way to expose your Domino data to none Notes systems or none Notes clients/applications, such as Oracle, .Net, Flex/Flash or mobile devices (e.g. Blackberry, iPhone). Basically to any framework that can consume web services.

This without requiring changes to any of your existing Domino applications (databases) and - equally important - without exposing your data directly to the world. The soapgate Q! database would typically reside on the public Domino server (located in the DMZ), accessing databases on the Domino server(s) in the private (firewalled) network.

The following chart illustrates the above:



The soapgate Q! web service runs as a Domino web user and hence supports all Domino authentication requirements to the respective databases that are accessed via the web service.

Installation and system requirements

soapgate Q! is developed for Lotus Domino 8.5

Installation

1. soapgate Q! does not come with an installer as there are only a few simple steps to “plug and play”
2. Copy the soapgate Q! template into the data directory of your Notes client (e.g. c:\program files\ibm\lotus\notes\data)
3. From the template create a new soapgate Q! database on your Domino server in the server's root (recommended) data directory.
4. Open the Notes Administrator client, connect to the respective Domino server and sign the soapgate Q! database with the server's ID-file.
5. Configure the soapgate Q! ACL accordingly. Recommended are following ACL entries:
   - -Default - (Unspecified) = No Access
   - Anonymous (Unspecified) = Editor (for public sites)
   - Anonymous (Unspecified) = No Access (for extranet scenarios)
   - <Your Server> (Server) = Manager
   - Username(s) (Person) = Editor (for extranet scenarios)
   - Groupname(s) (Person group) = Editor (for extranet scenarios)
6. The server that is to host soapgate Q! must run the Domino HTTP task. So if Domino is not the public HTTP server, but for instance IIS or an Apache etc., the Domino server has to be configured to run its HTTP task on a port other then port 80.This type of configuration of course effects firewall configuration requirements as well as the external systems accessing the soapgate Q! web service operations. For the latter the port has to be part of the calling URL.

What Is New

A number of significant improvements have been implemented in soapgate Q! 2.0, which can be categorized into the following main areas:

1. Improved performance
2. New security layer
3. New web service operations
4. Additional features for existing web service operations

Improved performance

We have put quite some efforts into those web service operations returning large data sets, such as dbcolumnX, dblookupX and dbview2xml. Our tests show a performance boost of up to 200% compared to version 1.0.

New security layer

Besides the fact that soapgate Q! has always adhered to all Domino security layers, we have added a new security layer which is specific to the concept of a data access API. In soapgate Q! 2.0 the network administrators can configure so called Database Access Profiles to define which databases, which of their design elements and which content is accessible through the API.

New web service operations

• dbClearFolder	Removes all documents from a folder
• dbDocAttachmentList	List of attachments contained in a given document
• **dbDownloadFile**	Download document attachment
• **dbGetDelStubs**	List of Notes Deletion Stubs (UNIDs of deleted documents)
• dbPutInFolder	Add (list of) documents to folder
• dbRemoveFromFolder	Removes (list of) documents from a folder
• **dbUploadFile**	Upload file as document attachment

Additional features for existing web service operations

• dbColumnX & dbLookupX	Additional parameters provide for the concept of paging
• dbLookupX	A modified-since parameter allows to set a cut-off date
• dbView2XML	Categorized views can be returned as flat XML instead of hierarchical XML; a new parameter “simple” can be used to omit view design information for higher performance

Community Edition License

With soapgate Q! 2.0 we are introducing the Community Server Edition license. The Community Server Edition license is absolutely free. **Please read the next chapter for more details.**

Licensing

The Community Server Edition of soapgate Q! IS FREE !!! It is an unlimmited user, single server licence (one server per Notes domain).

The Enterprise Server Edition requires a support contract.

Whilst soapgate Q! is not open source, Qkom.net and Flexdomino.net provide plenty of free and open source sample projects and library code for Flex developers and other developers, too.

To receive your Community Server Edition licence key please register with Qkom.net or Flexdomino.net. After the registration is completed a licence key is provided free of charge.

Win-Win through Fair Play

The idea of our Community Server Edition is to provide developers with a no cost solution to their companies or personal Domino IT requirements. We would also like to encourage the development of mobile applications for iOS, Android, QNX (Blackberry Playbook) and WebOS. However, we believe it is fair to ask you to make these applications FREE and Add-FREE (no advertising, except for yourself or your company that is).

If you develop "of the shelf" type applications that are distributed through any of the mobile device selling platform, such as the Blackberry Appworld, the Android Marketplace or the Apple App Store - to name but a few - and you intend to go commercial with your application, that includes applications that are FREE, but have revenue generating advertising included, we would like to have our share.

In return, if you develop applications utilising SoapgateQ! and your customers prefer to have support for SoapgateQ! (and your application) and this results in revenue on our end through the sale of the Enterprise Server Edition, we are more than happy to share the revenue with you.

To keep it simple we set the share at 20% both directions. So if you earn money through your SoapgateQ! enabled application, we would like to earn 20% of that and if we sell an Enterprise Server Edition to your customers we will pay 20% of our revenue. Fair and square.

Please refer to our distribution partner Qkom for more details.

Bug fixes & improvements in soapgate Q! 2.1

- dbviewftsearch() returned always the first found entry x-times (whereas x = number of found entries). This has been fixed.
- Field level API security was not always working as expected. This has been fixed.
- Some UI related issues in the Database Access Profile have been fixed.
- Similar to the Field Order & Prompts tab, the View Order & Display Names tab has been added to the Database Access Profile.
- Use of formula fields in the Database Access Profile
- Generic form fields in the Database Access Profile in the notation *ANY:fieldname* valid for all forms

Support

Qkom.net provides technical support for users of the Enterprise Server Edition or users with a support contract.

For users of the Communit Server Edition or without support contract we provide a technical user forum at.

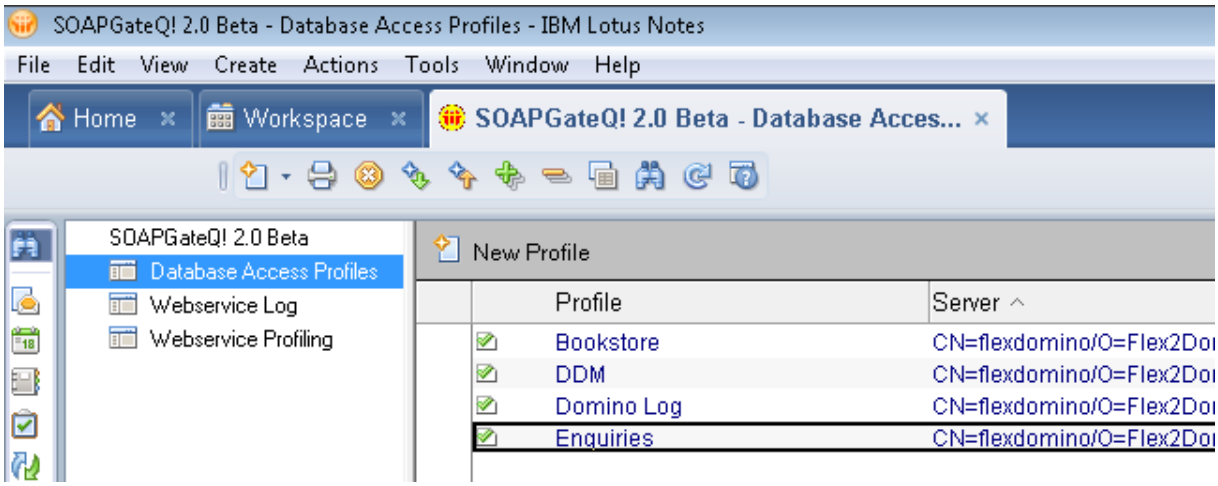
For all the above please refer to…

http://www.qkom.net/ibmdomino/soapgateq
http://www.flexdomino.net
http://flexdomino.blogspot.com

The soapgate Q! database

Navigation & Views

The soapgate Q! database is very simple in design and provides for the bare necessity to set up and configure the data access API, to monitor the web service utilization (log) and to run some basic maintenance. The navigator provides 3 basic views:



Database Access Profiles (new)

Database Access Profiles are part of the new security layer in soapgate Q! 2.0. It allows Domino administrators and or database managers to define what kind of access users is given through the data access API. This view lists all created profiles by name.

Please refer to the Database Access Profiles section for detailed information.

Webservices Log

Lists all debugging/log entries according to the debug/logging level set in the Configuration Profile (see Configuration & Maintenance).

Webservices Profiling

Lists Profiling information based on the last executed web service operation. The view shows each executed function with the total time in seconds and the total number of calls (see Configuration & Maintenance).

Configuration & Maintenance

The soapgate Q! database contains 3 agents under the actions, administrations menu for configuration and maintenance:

Clear Design Cache

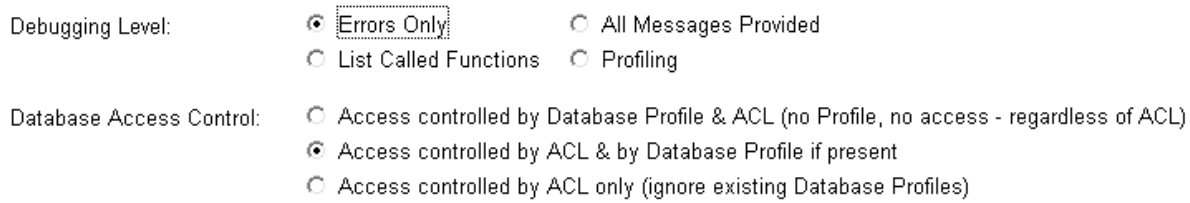
The soapgate Q! web services related to reading Notes View data do cache the view design in view specific profiles. The reading of the view design is required to provide formatting information. However, a view design cannot be read through normal user access privileges, but only with at least designer level. Most users do not have the access level and hence soapgate Q! contains an agent running with server privileges to read the design and cache it in a database profile.

The Clear Design Cache agent allows deleting these profiles from the database. A developer would usually run this agent during development time when the view design is subject to frequent change. The cached design expires after 24 hours, after which the view design will be read in again.

Note: a user executing the agent must have at least Editor Rights and the Delete Documents option tick-marked.

Edit Config Profile

This agent opens a configuration profile, which allows the configuration of a logging level and the method in which the new security layer is implemented:



- **Debugging Level**

  Errors Only

  Only internal code errors and a few defined validation errors are logged.

  List Called Functions

  In addition to error logging, every single function called during the web service operation is logged. However, some of the functions only for the first call.

  All Messages Provided

  Logs additional information such as parameters (the web service operations where called with), return values and other eventually important debugging information.

Profiling

All above logging options are disabled - except for errors -, and only one log entry is produced showing all functions executed during the web service operation, with total calls per function and total time consumed.

Profiling itself is time consuming and therefore the recorded times are inflated. The overall consumed time (total) is therefore not very informative. However the values give an indication as to which of the called functions during a web service operation are potential candidates for code optimization. As soapgate Q! is not open source, this feature is probably only of use to flexdomino.net.

- **Database Access Control** (new)

This option is related to the new security layer provided with the Database Access Profiles.

Note: the security level chosen is on top of the Domino security for authenticated (and none authenticated users). It does not replace it.

Access controlled by Database Profile & ACL (no Profile, no access - regardless of ACL)

This is the highest level of security for the data access API. Each database accessed through the API must have a profile set up. A Database Access Profile controls who can access the database and what design and data (outlines, views, forms, fields).

Access controlled by ACL & by Database Profile if present

This setting provides a medium level of security. If a profile exists for a database and the user is in the readers list of this profile, the user is limited in its access by whatever is configured in the profile. If no profile exists or is assigned to the user, access control is determined by ACL (Access Control List) of the accessed database.

Access controlled by ACL only (ignore existing Database Profiles)

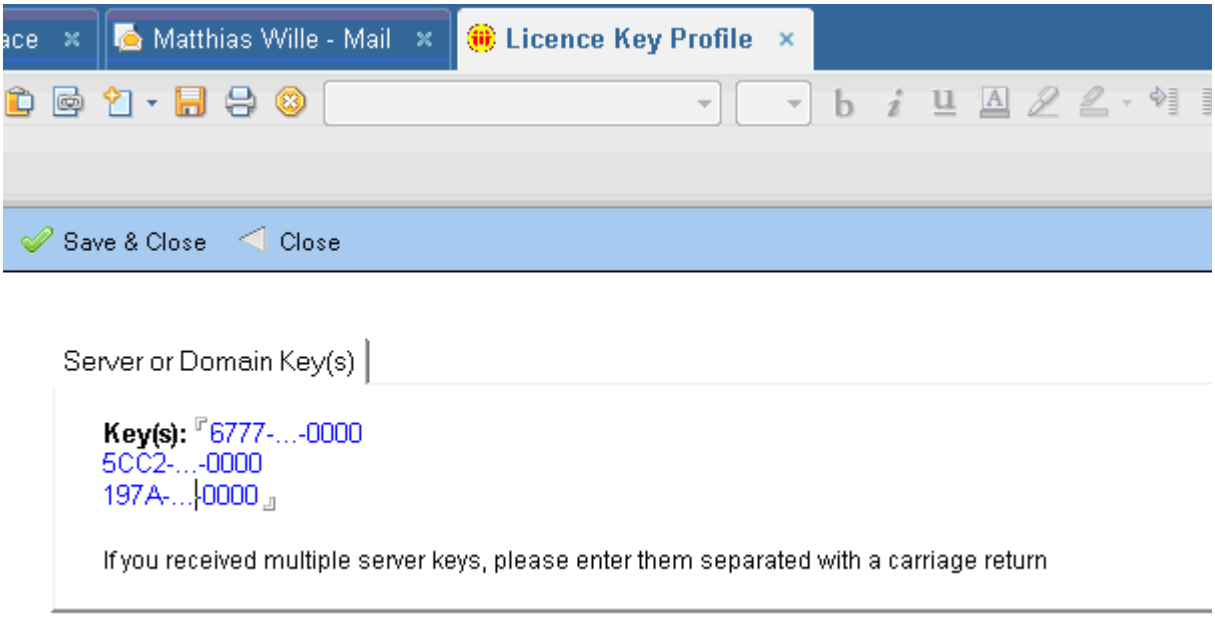
This setting provides the lowest level of security. It will ignore all existing Database Access Profiles. Access control is determined by ACL (Access Control List) of the accessed database.

Edit License Key Profile

This document holds the license key(s) for the soapgate Q! database.

Server or Domain Keys

Simply copy and paste all provided keys into the Key(s) field separated by a carriage return.



Database Access Profiles (new)

The data access API of soapgate Q! is designed to give external systems/networks/users access to data traditionally located inside a corporate internal networks protected by firewalls and other security measures. For instance through applications for mobile devices, providing remote access to the corporates operational systems.

A Database Access Profile collects all the design information about a database that is relevant for the data access API of soapgate Q! to control what design and data can be accessed from the database, and by whom.

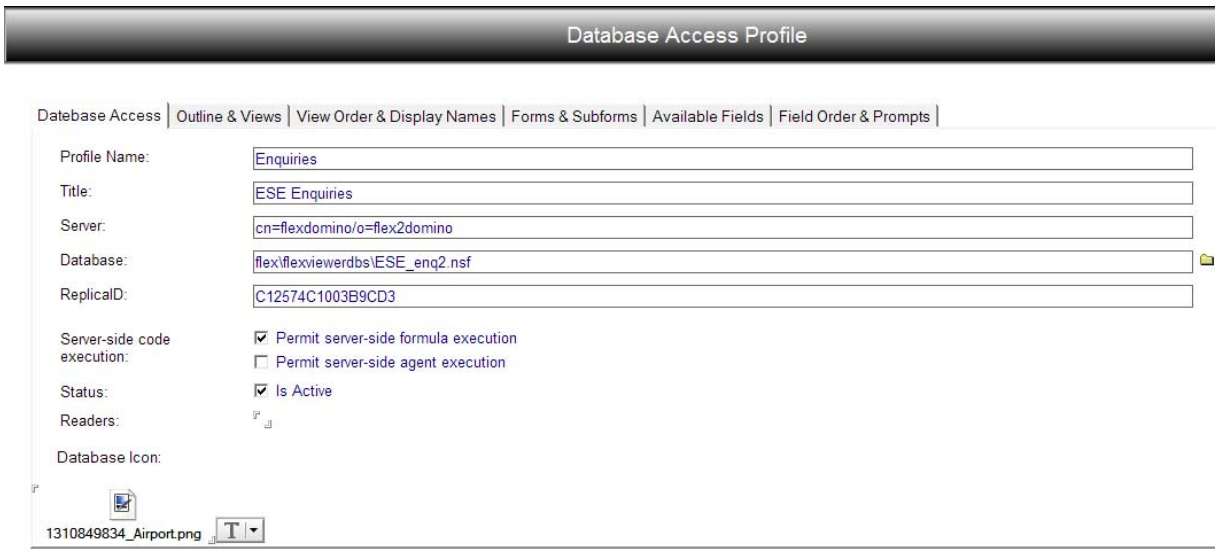
This additional security layer defines the data accessibility from the outside or – to turn this around - what design and data will rather not be accessible, regardless of the individual rights given by the implemented Domino security.

Hence, what a Database Access Profile allows is to restrict a user's access to a subset of data the user would have access to from inside the network.

The Database Access Profile is organised in following sections (tabs)

- **Database Access** – Allows the selection of the database, the profile status and reader access
- **Outline & Views** – Allows to select a specific outline and or a number views for API access
- **View Order & Display Names** – Defines the order in which views and folders are returned and the prompts with which they should be displayed.
- **Forms & Subforms** – Allows to select a number of forms and subforms for API access
- **Available Fields** – Allows to select a number of fields for API access
- **Field Order & Prompts** – Defines the order in which field values are returned and the prompts with which they should be displayed.

Database Access



Profile Name

The Profile Name must be unique, though no validation is currently in place.

Database (Title, Server, Replica ID)

The database can be selected using the Notes file open database dialog. All fields will be filled automatically. Once the database is selected the database design is read in and the lists of available views, forms and fields are displayed.

Server-side code execution

Only if *Permit server-side formula execution* is enabled, will those web service operations providing for the transmission of Notes Formula actually execute the latter. Otherwise any transmitted formula will simply be ignored.

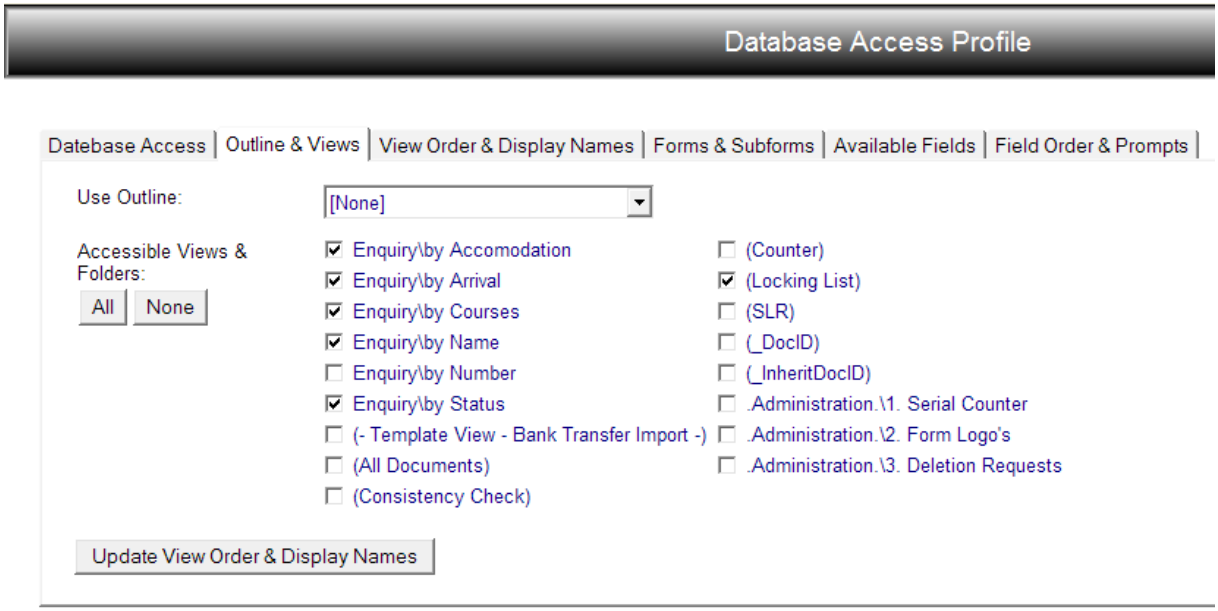
Permit server-side agent execution must be enabled to be able to use the *dbCallAgent* web service operation.

Permit server-side agent execution must also be enabled when creating computed fields in the Fields Order & Prompts tab.

Database Icon

To create a database browser (a sort of mobile Notes Workspace) it is possible to attach database icon in this rich text field.

Outline & Views



Use Outline

Selecting an outline limits access to those views and folders linked in the outline. It is not required to select any views in the Accessible Views & Folders field when selecting an outline, though it is possible to add views or folders not linked in the outline.

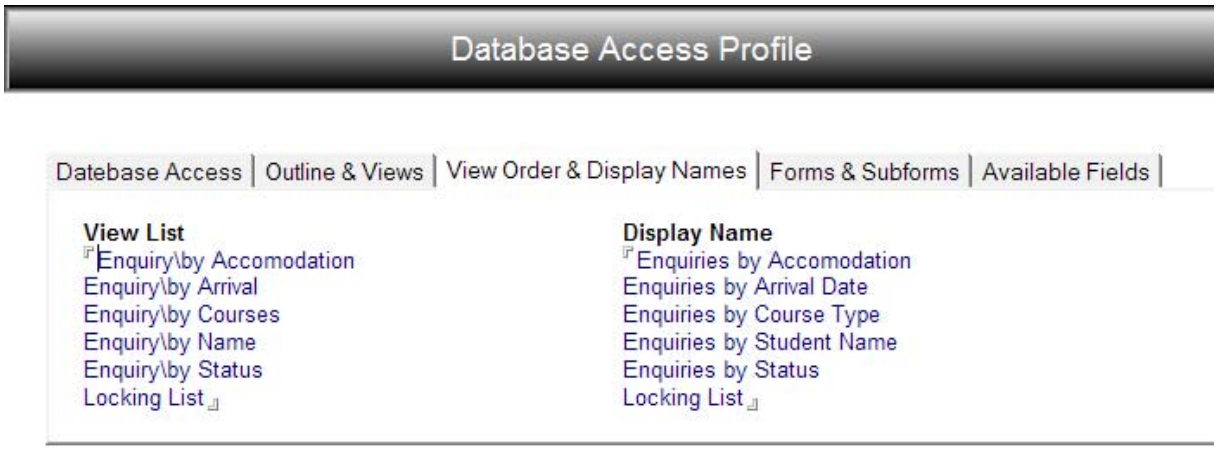
Note: this feature is not yet implemented.

Accessible Views & Folders

Only the views and folders selected in the list are accessible through the data access API. If the View List and Display Name fields in the View Order & Display Names tab have not been filled yet, a simply document refresh (F9 key) will add all selected views. This is a one time auto add feature. After that, you are required to use the Update View Order & Display Names button. Clicking the latter will overwrite the content of the View List and Display Name fields. So use with caution as you might loose your custom order and display names.

Note: All web service operation calls that request view data, but refer to document fields instead of view or folder columns (for instance when calling dbcolumnx), require the fields listed in the Field Order & Prompts table in the notation *view_access.<fieldname>.* Once field access is granted the *view_access.<fieldname>,* the field is accessible through all API enabled views & folders.

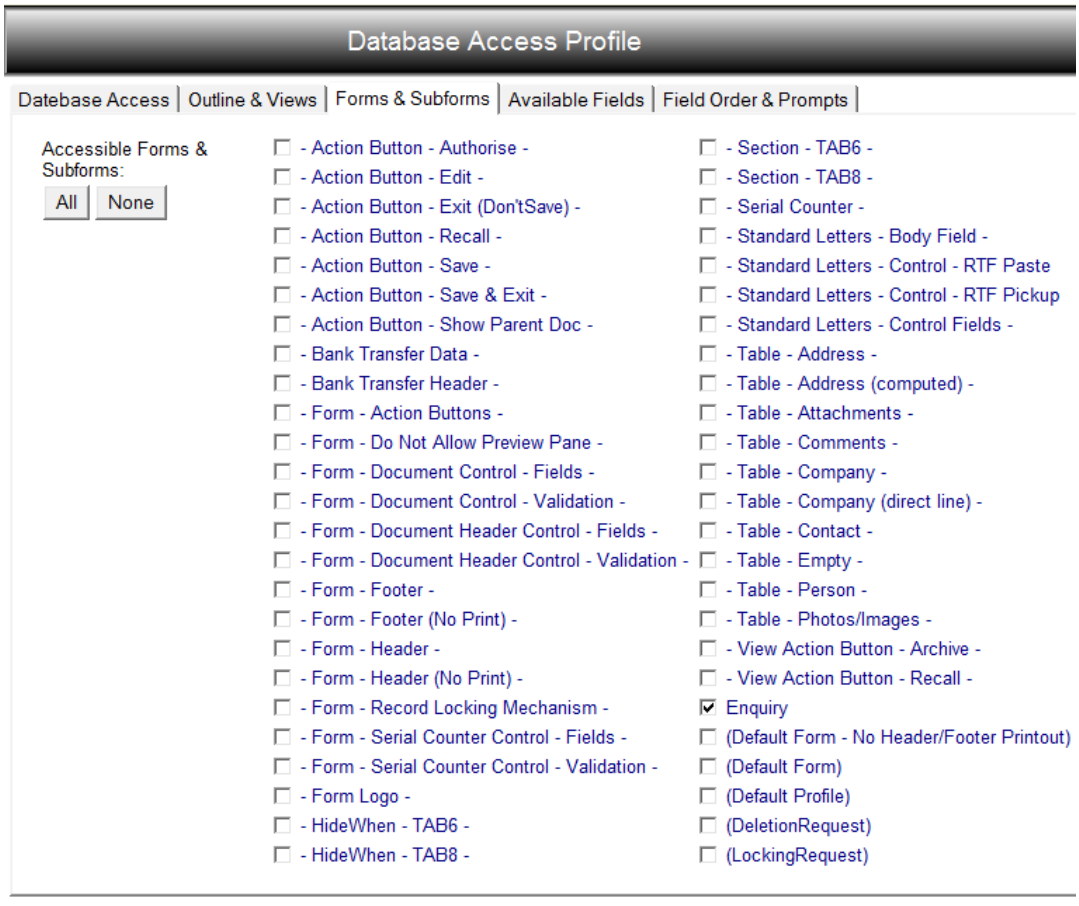
View Order & Display Names (new in 2.1)



In this tab you can define the custom order and display names of the views and folders accessible through the API. This is only of relevance if you provide an UI interface that shows some kind of navigator in whatever platform you develop your application.

Simply use cut and paste to create the view order as desired. Make sure for every view entry there is a corresponding display name. One view or display name respectively per row.

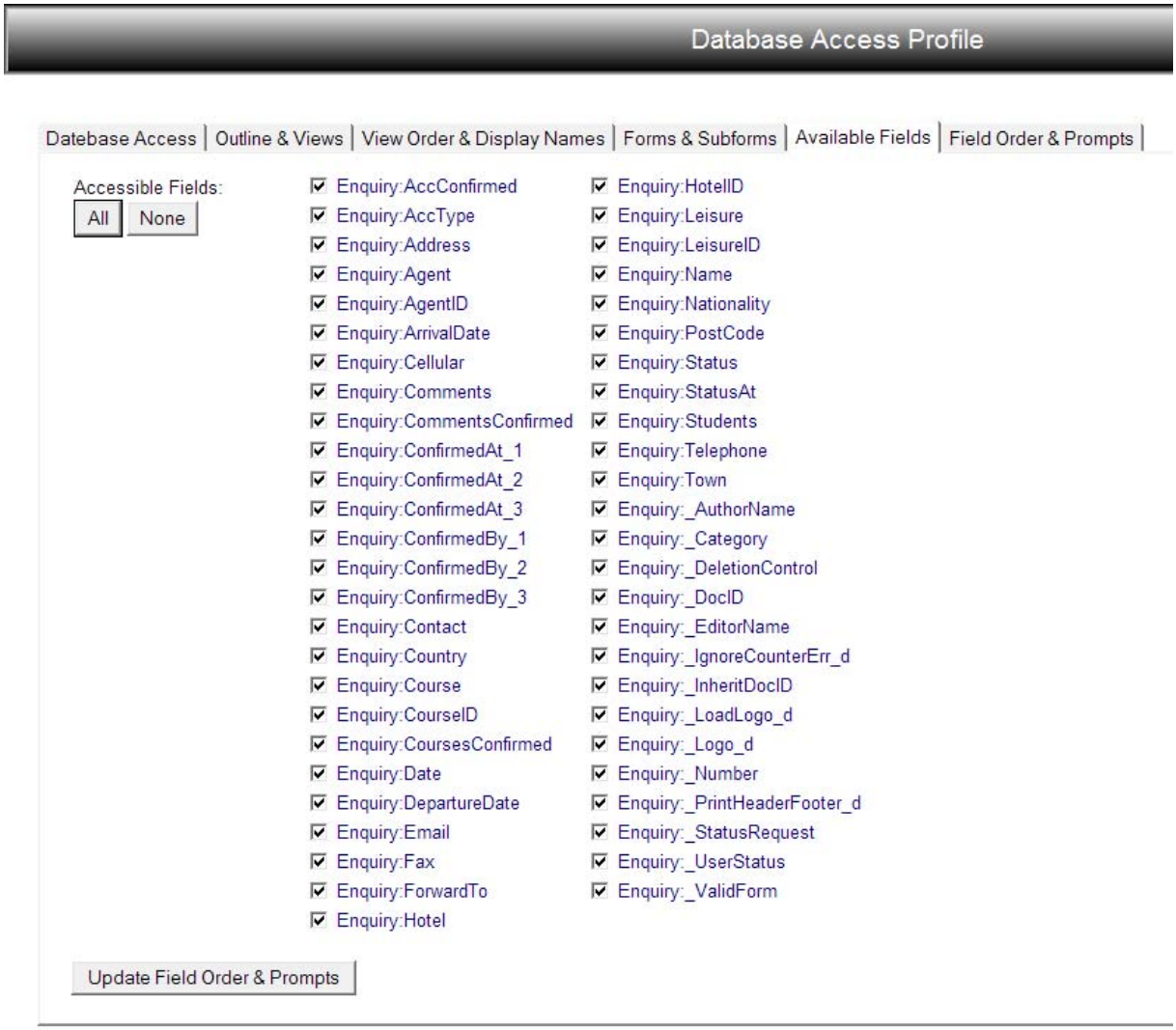
Forms & Subforms



Only documents based on the forms (refers to *form* field on document) selected in this list are accessible through the data access API. The also listed subforms are not relevant for the access to documents, however, selecting the latter allows adding fields contained in the subforms to be added to the Available Fields list.

Note: If a form contains named subform(s), any field contained in the subform(s) is also listed in the form's field list (form design document), hence it is not required to select the subform to add the subform fields. However, if the form contains computed subforms, then fields contained in these subforms are obviously not listed in the form's field list, hence the subform must be selected to add the fields to the Available Fields list.

Available Fields



The Available Fields list shows the fields of all selected forms & subforms in the notation *formname:fieldname* (or *subform:fieldname* respectively).

If the Selected Fields and Field Prompts fields in the Field Order & Prompts tab are not filled, the first document refresh after selecting fields in the Accessible Fields list will add them to the Field Order & Prompts table. This is a one time auto add feature. After that, you are required to use the Update Field

Order & Prompts button. Clicking the latter will overwrite the content of the Selected Fields and Field Prompts fields. So use with caution as you might loose your custom order and prompts and any computed fields you may have added to the list.

Field Order & Prompts

Only fields listed in this table are available through the data access API. To allow simplified input forms on mobile devices, the respective web service operations returning form field content will return the field values in the order they appear in this table. The order of the requested field list in such web service operations (fieldlist parameter) will be ignored. So will any fields be ignored that are requested, but not listed in this table.

Generated documents fields without corresponding form fields

If documents contain fields generated e.g. by an agent that are not part of any form or subform, simply add an entry in the same notation as shown above: *formname:fieldname*.

Generic fields that are available on many (all) forms **(new in 2.1)**

If forms or documents contain generic fields to be made available on all accessible documents/forms, the field can be added in a simpler notation: *ANY:fieldname*. This will make the field accessible in all forms that are made available.

Accessing fields through view related web service operations

All web service operations that request view data, but refer to document fields instead of view or folder columns, require the fields listed in the Field Order & Prompts table in the notation *view_access.<fieldname>.* For instance *view_access.Students*.

Computed fields (server side) **(new in 2.1)**

Provided server side code execution is enabled, it is possible to add formulae instead of a field name to the Selected Fields list. The above screen shot shows a computed field with following formula:

```
Enquiry:@"**********************************************************"
```

Of course more complex formulae are possible. For instance if a document has a computed for display field with a formula difficult to repeat in a different platform, simply add the same formula of that computed for display field to the Selected Field list:

```
Activity:@SpaceK:=DiskSpace/1024;@V2If(SpaceK<1024;@Text(@Round(SpaceK;1))+"
Kbytes";@Text(@Round((SpaceK/1024);0.1))+" Mbytes")
```

The above sample is taken from the log.nsf Activity form.

Note:

- The @ symbol indicates the use of formula, any leading @ of a formula must be included: Enquiry:@@Now.
- The formula must be stripped of any carriage returns.
- A blank Prompt is not possible, simply add on space character
- It is possible to add custom seperators to the Selected Fields and Field Prompts fields to make the start of the next form in the list more visible. Any string of characters that does not contain a ":" (colon) will do...

```
Activity:@"**********************************************************"
Activity:Managers
Activity:@@V2If(@IsAvailable(ViewTitles);ViewTitles+@Char(9)+@Text(@Round(ViewSiz
es/1024;1));"No views")
----------
Events:Server
Events:StartTime
Events:FinishTime
Events:@@Explode(@Implode(@If(@IsAvailable(Eventlist);EventList;@IsAvailable(Eventsli
st);EventsList;Events);"~~");"~";1)
----------
Session:Server
```

```
Managers
Views
----------
Server
Start Time
Finish Time
Events
----------
Server
StartTime
FinishTime
UserName
```

- Make sure the list of Selected Fields has the same number of entries as the list of Field Prompts. The (only) entry seperator is the carriage return.

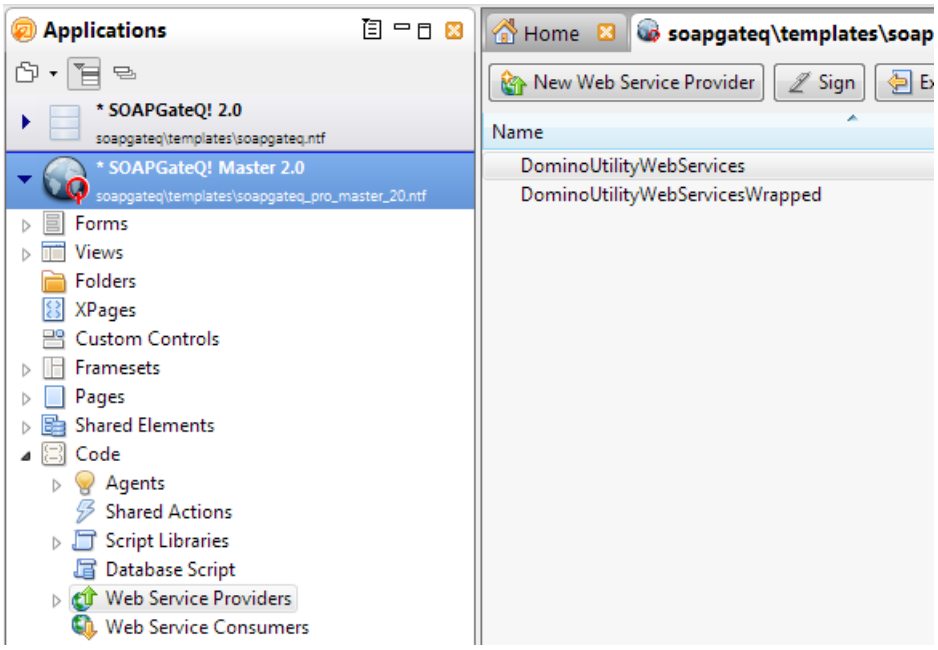
Limitations and considerations of the API security layer

- In version 2.1 column level security is not yet implemented, essentially, once a view is made available, all data shown in the view is available.
- Enabling server-side formula execution provides the means to overwrite the security implemented with the Accessible Forms & Subforms as well as Accessible Fields selection. Currently there is no distintion between formulae coming from a Database Access Profile and other sources.
- The FORM field on the Notes document decides about the accessibility of fields, any view Form Formula will be ignored.

Consuming soapgate Q!

Creating a soapgate Q! consumer for Adobe® Flash® Builder™ 4

Adobe® Flash® Builder™ 4 provides a tool to import a WSDL file and hence to create the web service consumer classes in an automated fashion. However, it cannot swallow any type of Domino web service, but only wrapped RPC. The same applies also for the .Net environment. For this reason the soapgate Q! database contains identical web services for all of the by Domino supported types:



The DominoUtilityWebService web service is of type doc/literal RPC and can be consumed by native applications developed for Blackberry devices and also with Flex provided the consumer classes are developed without using the WSDL import wizard. The DominoUtilityWebserviceWrapped web service is of type wrapped RPC and can be consumed by .Net and Adobe® Flash® Builder™ 4 (using the WSDL import wizard).

Note: web services with the OEM name extension will be used 3rd party OEM implementations of soapgate Q! and can be ignored for the time being.

The following method to generate the web service wrapper code by importing the soapgate Q! WSDL-file in Adobe Flex Builder is similarly available in many other IDE's and should just be understood as an example.

1. Open Adobe® Flash® Builder™ 4
2. Create a new Flex project
3. In the menu select Data->Connect to Web Service…
4. In the 2nd step enter following WSDL URL: http://domino.flexdomino.net/SOAPGateQ.nsf/DominoUtilityWebServicesWrapped?wsdl and click Next

5. In the 3rd step the provided web service operations are listed. Click Finish to complete the import process.

Please refer to the Adobe® Flash® Builder™ 4/Adobe SDK manuals on how to use the generated web service consumer classes in your Flex projects.

Although it is perfectly possible to consume the soapgate Q! web services in Flash Builder 4 using the above described method, it is much simpler to use the Flex/Flash components for soapgate Q!, which have been development without using the WSDL import wizard of Adobe® Flash® Builder™ 4 and therefore the doc/literal RPC web service (DominoUtilityWebService) is utilized. The Flex/Flash components for soapgate Q! are open source and licensed under MIT.

API Reference

NOTE:

- Domino web service operations are always defined upper case (see WSDL).The proper case writing in this manual is for the purpose of an easier readability.
- The *dbname* (database name) parameter must contain the filepath to the database. Using the replica ID is not supported.

If Database Access Profiles are enabled and a matching profile exists

- Denied access to forms and views causes the web service operation to fail and return an error string: [Error.APISecurity] Access to requested view is denied.
- Denied access to fields or server side formula execution does NOT cause the operation to fail, but the requested data is replace by ”******”

dbACL

Web service implementation of the NotesACL class (read only)

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)

Return value:

String:XML

dbCallAgent

Web service implementation of the Domino class property NotesAgent.Run().

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
agent	String	Name of agent to be executed
paramXML	String:XML	Any XML formatted string which will be attached to an agent context document in a pre-defined field called *agentcontextXML*
noteID	String	Note ID to an agent context document. If blank, the default agent context profile is used (profile name = ”SoapGateQ”, profile key = effective user name)

Return value:

String:XML

The return value is either “” (null string) or taken from the agent context document from the pre-defined field *agentreturnXML*.

dbClearFolder (new)

Web service implementation of the Domino class method NotesView.AllEntries.RemoveFromFolder (where NotesView is a view of type folder).

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
folder	String	Name of the folder to be emptied

Return value:

String

The return value is either "1" (true) if the operation was successful or it contains an error string.

dbColumn

Web service implementation of the Notes Formula function @dbcolumn().

Parameters:

cache	Boolean	If false Domino will update its cache from the data on file
srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
viewname	String	Name or alias of the view, which data is to be returned
colnumber	Integer	Index of view column which data is to be returned starting from 1 for the first column in the view
mvalsep	String	Multi-value separator

Return value:

Array: Any

Serialization:

Because of the required data serialization for multi-value columns, such values will be converted to type string, each value separated by the character passed on in mvalsep.

See appendix for more information.

dbColumnX (changed)

Web service implementation of the Notes Formula function @dbcolumn() with two extended features:

- returns one or more columns
- returns columns and or fields
- always returns the view entry number (last-1 returned column)
- always returns the noteID or UNID (last returned column)

Note:

- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *retUNID*.
- the operation adds yet another column containing the view entry number of the document
- two new parameters – *startpos* and *rows* – have been added, to allow paging

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
viewname	String	Name or alias of the view, which data is to be returned
colfields	Array	List of column numbers and or fieldnames
mvalsep	String	Multi-value separator
retUNID	Boolean	dbColumnX 2.0 always returns either the noteID or UNID of the documents. If retUNID is true the UNID is returned, otherwise the noteID.
rowformat	Boolean	If true data is returned by row rather than by column (see Return value)
startpos	String	View entry number (string) as a means of an offset from which to read the next batch of documents. If blank, reading starts from the first document in the view.
rows	Integer	Number of documents to be read from *startpos.* If *rows* is zero, all remaining documents beginning from *startpos* are read.

Return value:

Array: Any

Serialization:

Due to data serialization the returned value is a single dimensioned array. The row format parameter determines the result of the serialization. See appendix for more information.

dbDeleteDoc

Deletes a document from file.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
docUNID	String	Universal ID or Note ID of the document that is to be deleted
force	Boolean	If true, the document will be deleted even if it is currently edited by another user

Return value:

String: “true”, “false”

dbDocAttachmentList (new)

Returns a list of attachments stored in a document or richtext field within a document.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
noteID	String	Universal ID or NoteID of the document which contains the attachment
rtfield	String	Richtext field containing the attachment. If the attachment is not bound to a Richtext field, leave the value blank (“”)

Return value:

String: XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<attachments>
        <file>
                <name>dinosaurs.JPG</name>
                <source>dinosaurs.JPG</source>
                <size>32637</size>
        </file>
</attachments>
```

dbDownloadFile (new)

Allows downloading of files as a Base64 encoded (string) stream.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
noteID	String	Universal ID or NoteID of the document which contains the attachment
rtfield	String	Richtext field containing the attachment. If the attachment is not bound to a Richtext field, leave the value blank (“”)
filename	String	Name of attachment (filename excluding filepath, e.g. mypicture.jpg)
useIBMencode	Boolean	IBM’s undocumented internal NotesStream.ReadEncoded() method has a bug which will not be fixed before version 8.5.3. This bug effects larger files (>1MB) http://flexdomino.blogspot.com/2011/03/fixed-lsxsdlss.html Because of that, we rewrote the older base64 encoding algorithm in LSXSD.LSS, as it too had issues related to performance. If useIBMencode is *false*, the performance improved old base64 encoding is used, otherwise the new IBM internal method. **Note: once the issue with .ReadEncoded() is fixed, this parameter should be set to *true*, as the new encoding method is a multitude times faster than the old LS one.**

Return value:

String: XSD_BASE64BINARY stream containing the base64 encoded file.

dbEffectiveRights

Web service implementation of the NotesDatabase.Queryaccess method. The method is called for the authenticated user (session.Effectiveusername).

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)

Return value:

String:”0”...”6”

See appendix for access level table.

dbFTSearch (changed)

Web service implementation of the Domino class method NotesDatabase.FTSearch() with extended features:

- returns one or more document fields
- always returns the noteID or UNID (last returned column)

Note:
- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *returnUNID*.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
query	String	The full-text query. Please refer to the Lotus Notes Help for syntax information
maxdocs	Integer	The maximum number of documents you want returned from the query. Set this parameter to 0 to retrieve all documents (up to 5,000) that meet the search criteria
sortoption	Integer	Optional. Sorting options. (64) sorts by document date in ascending order. (32) sorts by document date in descending order. (1543) sorts by document creation date in ascending order. (1542) sorts by document creation date in descending order. (8) sorts by relevance score (default)
otheroptions	Integer	Optional. Search options. Combine options by adding (8192) includes Domino databases. (16384) searches for related words. Need not be an exact match. (4096) includes files that are not Domino databases. (512) uses stem words as the basis of the search. (1024) uses thesaurus synonyms
colfields	Array	List of column numbers and or fieldnames
mvalsep	String	Multi-value separator
returnUNID	Boolean	If true it returns the UNID of the document in addition to the requested fields, otherwise the NoteID
rowformat	Boolean	If true data is returned by row rather than by column (see return value)

Return value:

Array: Any

Serialization:

Due to data serialization the returned value is a single dimensioned array. The row format parameter determines the result of the serialization. See appendix for more information.

dbGetDelStubs (new)

This operation returns the deletion stubs for the given database. This includes the following note classes:

NOTE_CLASS_DOCUMENT	"Document"
NOTE_CLASS_INFO	"Help-about"
NOTE_CLASS_FORM	"Form"
NOTE_CLASS_VIEW	"View"
NOTE_CLASS_ICON	"Icon"
NOTE_CLASS_DESIGN	"Design collection"
NOTE_CLASS_ACL	"ACL"
NOTE_CLASS_HELP_INDEX	"Help index"
NOTE_CLASS_HELP	"Help-using"
NOTE_CLASS_FILTER	"Filter"
NOTE_CLASS_FIELD	"Field"
NOTE_CLASS_REPLFORMULA	"Replication formula"
NOTE_CLASS_PRIVATE	"Private design"

OR

"Unknown"

As Notes deletion stubs get purched according to the set database properties, the same considerations for this operation's usage apply as for the Notes Client to Domino server, or server to server replication.

The web service operation is based on some C-API code posted on IBM's website:

http://www-10.lotus.com/ldd/nd6forum.nsf/0/8210fa46540ecbbf852572b40044bb3e?OpenDocument

Note: this web service operation works only on Domino for Windows

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)

Return value:

String:XML

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<stubs>
	<stub>
		<DBID>49CDFC9F:F21653E7</DBID>
		<NoteClass>Document</NoteClass>
		<NoteID>EEA</NoteID>
	</stub>
	<stub>
		<DBID>E7BA3AA5:9107E781</DBID>
		<NoteClass>Document</NoteClass>
		<NoteID>EF2</NoteID>
	</stub>
</stubs>
```

dbGetFieldTypes

Returns the data types for a given list of fields contained in a given form.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
form	String	Name or alias of the form which field list is to be returned
fields	Array	List of fieldnames which field types are to be returned
mvalsep	String	Multi-value separator

Return value:

Array:Integer

Data types:

See appendix

dbGetFormFields

Returns the list of fieldnames and types contained in a specified form.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
form	String	Name or alias of the form, which field list is to be returned
includeLSF	Boolean	If true includes Lotus $ (system) fields

Return value:

Array: String and Integer

fieldname1
fieldname2
fieldname3...
fieldtype1
fieldype2
fieldtype3...

Serialization:

Due to data serialization the returned value is a single dimensioned array. The array starts with the list of field names and continues with the list of data types.
See appendix for more information.

Data types:

See appendix

dbGetMailInfo

Web service implementation of the Domino class method NotesDirectory.Getmailinfo(). The latter provides for three parameters of which only the first one (username) is supported, the second and third parameter are fixed to true and false respectively.

Parameters:

username	String	Name of the Domino server on which the source database resides

Return value:

Array: String

Elements returned are as follows:

MailServer	Home mail server for the specified person
BuildNumber	String representation of the build number of the specified person's mail server, for example,"303"
DominoVersion	String representation of the Domino version of the specified person's mail server, for example," Build V80_07042006NP
MailFile	Mail file for the specified person
ShortName	Short form of the specified person's name
MailDomain	Notes Domain of the specified person's mail address
UserName	First entry in the list of user names honoured for the specified person
InternetMailAddress	Internet mail address for the specified person
OutOfOffice	Out of Office service type."1" indicates Agent,"2"indicates Service

dbLookup

Web service implementation of the Notes Formula function @dblookup().

Parameters:

cache	Boolean	If false Domino will update its cache from the data on file
srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name or alias of the view, which data is to be returned
keys	Array	A value to be matched against the first sorted column in the view
colnumber	Integer	Number of a column in the view starting at1.Ifa column number is used, the return values are taken from this column in the view
fieldname	String	Name of an item in the documents in the view. If an item name is used, the return values are taken from this item in the documents from which the view data is drawn
mvalsep	String	Multi-value separator
failsilent	Boolean	returns "" (null string) instead of an error if the key cannot be found
partialmatch	Boolean	returns a match if the key matches the beginning characters of the column value
returnUNID	Boolean	returns the UNID of the document instead of afield or column value

Return value:

Array: Any

Serialization:

Because of the required data serialization for multi-value column or fields, such values will be converted to type string, each value separated by the character passed on in mvalsep. Otherwise data types remain intact. See appendix for more information.

dbLookupX (changed)

Web service implementation of the Notes Formula function @dblookup() with two extended feature:

- returns one or more columns
- returns columns and or fields
- always returns the view entry number (last-1 returned column)
- always returns the noteID or UNID (last returned column)

Note:

- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *retUNID*.
- the operation adds yet another column containing the view entry number of the document
- two new parameters – *startpos* and *rows* – have been added, to allow paging

Parameters:

cache	Boolean	If False Domino will update its cache from the data on file
srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name or alias of the view, which data is to be returned
keys	Array	A value to be matched against the first sorted column in the view
colfields	Array	List of column numbers and or fieldnames to be returned
mvalsep	String	Multi-value separator
failsilent	Boolean	returns "" (null string) instead of an error if the key cannot be found
partialmatch	Boolean	returns a match if the key matches the beginning characters of the column value
retUNID	Boolean	dbColumnX 2.0 always returns either the noteID or UNID of the documents. If retUNID is true the UNID is returned, otherwise the noteID.
rowformat	Boolean	If true data is returned by row rather than by column (see Return value)
startpos	String	View entry number (string) as a means of an offset from which to read the next batch of documents. If blank, reading starts from the first document in the view.
rows	Integer	Number of documents to be read from *startpos.* If *rows* is zero, all remaining documents beginning from *startpos* are read.

Return value:

Array: Any

Serialization:

Due to data serialization the returned value is a single dimensioned array. The row format parameter determines the result of the serialization.

See appendix for more information.

dbOutlineViews

dbOutlineViews extracts a list of views and folders referenced in a given Notes outline. This includes views and folders that are referenced via a formula.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
outline	String	Name of the outline which view references are to be extracted.
excludeHidden	Boolean	If true, hidden entries are ignored
excludeWeb	Boolean	If true, entries hidden for web use are ignored
excludeClient	Boolean	If true, entries hidden for client use are ignored
keepHierarchy	Boolean	If true, the outline's parent child entry hierarchy is reflected in the returned XML, otherwise the XML is a flat view list.

Return value:

String: XML

```
<views>
<node label="BooksFlat" data="books.flat"/>
<child  label="BooksHierarchical" data="books.hier">
<node label="BooksXML" data="books.xml"/>
</child>
</views>
```

The label attribute contains the view name, the data attribute contains the alias or name of the view or folder.

dbPutInFolder (new)

Web service implementation of the Domino class method NotesDocumentCollection.PutInFolder.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
foldername	String	Name of the folder to add the documents to
doccol	Array	List of documents (UNIDs or NoteIDs) to be added to the folder
createonfail	Boolean	If true will create the folder if it does not exist

Return value:

String

The return value is either "1" (true) if the operation was successful or it contains an error string.

dbReadDocFields

Returns the values and data types for the list of given fields of a given document.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Path name of the database (relative to the root data directory of the server)
viewname	String	Name or alias of the view, which data is to be returned
docUNID	String	Universal ID of the document which field values are to be returned
fields	Array	List of fields which values are to be returned
mvalsep	String	Multi-value separator

Return value:

Array: Any

fieldvalue1
fieldvalue2_a~fieldvalue2_b~... *
fieldvalue3...
datatype1
datatype2_a~datatype2_b~... *
datatype3...

* Serialization:

Due to data serialization the returned value is a single dimensioned array. The array starts with the list of field values and continues with the list of data types. For those field values that represent a Notes multi-value field, the multi-value content is serialized using the *mvalsep* character. The web service consumer must convert these back to arrays. The recommended (default) multi-value separator is “~” (tilde).

All field values are converted to string. This complies with the way the Notes Client UI document works. Whilst edited all fields in a Notes document are edited as text. Default and custom validations control the users input.

Similar to this concept the web service consumer, for instance a Flex form must have input formatters and validations implemented. Alternatively the consumer must convert the returned string data in accordance with the also provided data type information into whatever is required.

The correlating web service operation dbSaveDocFields converts the string data back into the respective data types required in the used Notes form for the document being modified.

Note: Multi-value fields in Notes cannot store mixed data types and therefore even for multi-value fields the datatype return could be a single value rather then a tilde (default) seperated datatype string. However, it is somewhat easier to handle data and datatype both being returned as an array.

Data types:

See appendix

dbReadProfileFields

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
profname	String	The name or an alias of the profile form
profkey	String	The unique key associated with the profile document
fields	Array	List of fields which values are to be returned
mvalsep	String	Multi-value separator

Return value:

Array: String

dbRemoveFromFolder (new)

Web service implementation of the Domino class method NotesDocumentCollection.RemoveFromFolder.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
foldername	String	Name of folder from which to remove the documents
doccol	Array	List of documents (UNIDs or NoteIDs) to be removed from the folder

Return value:

String

The return value is either "1" (true) if the operation was successful or it contains an error string.

dbRenderDoc

This feature is not yet implemented.

Renders a Notes-document into HTML or MIME

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
docUNID	String	Universal ID of the document which is to be rendered
noteID	String	Alternatively the NoteID
form	String	Name of form used for rendering the document
noCache	Boolean	True or false. Not yet supported parameter

Return value:

String: HTML/MIME

dbRowX (changed)

Returns column and or field values for a single view row.

- returns one or more columns
- returns columns and or fields
- always returns the view entry number (last-1 returned column)
- always returns the noteID or UNID (last returned column)

Note:
- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *retUNID*.
- the operation adds yet another column containing the view entry number of the document
- two new parameters – *startpos* and *rows* – have been added, to allow paging

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name or alias of the view, which data is to be returned
colfields	Array	List of column numbers and or fieldnames (for the single row) to be returned
mvalsep	String	Multi-value separator
retUNID	Boolean	dbColumnX 2.0 always returns either the noteID or UNID of the documents. If retUNID is true the UNID is returned, otherwise the noteID.
docUNID	String	Universal ID of the document representing the single view row to be returned

Return value:

Array: Any

Serialization:

Because of the required data serialization for multi-value column entries, such values will be converted to type string, each value separated by the character passed on in mvalsep. Otherwise data types remain intact.

See appendix for more information.

dbSaveDocFields

Updates field values in a given document or saves values in a new document.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
docUNID	String	Universal ID or Note ID of the document which field values are to be returned
fields	Array	List of fields which values are to be saved
types	Array	List of corresponding data types. Please refer to the appendix for the Notes field data types
values	Array	List of field values. If individual field values are multi-value then these values must be themselves send as a serialized string using the mvalsep character for separation. The Flex/Flash libraries for soapgate Q! will do this conversion automatically. For which reason in Flex the developer deals with arrays only, data serialization is transparent
mvalsep	String	Multi-value separator
form	String	Name of form used for new documents
compWF	Boolean	If true runs the Notes Document. ComputeWithForm() method on the server before the document is saved to file
saveRPC	String	Runs a Notes agent on the server before the document is saved file. This is the equivalent to the WebQuerySave event when submitting Notes form

Return value:

String

UNID/NoteID of the document amended or created (depending on parameter docUNID)

dbSaveProfileFields

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
profname	String	The name or an alias of the profile form
profkey	String	The unique key associated with the profile document
fields	Array	List of fields which values are to be saved
types	Array	List of corresponding data types. Please refer to the appendix for the Notes field data types
values	Array	List of field values. If individual field values are multi-value then these values must be themselves send as a serialized string using the mvalsep character for separation. The Flex/Flash libraries for soapgate Q! will do this conversion automatically. For which reason in Flex the developer deals with arrays only, data serialization is transparent
mvalsep	String	Multi-value separator

Return value:

String: UNID

Universal ID (UNID) of the profile that has been amended or created.

dbSearch (changed)

Web service implementation of the Domino class method NotesDatabase.Search() with extended features:

- returns one or more document fields
- always returns the noteID or UNID (last returned column)

Note:
- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *returnUNID*.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
formula	String	A Notes @function formula that defines the selection criteria
datetime	String	A cut off date. The method searches only documents created or modified since the cut off date. Specify “” (null string) to indicate no cutoff date.
maxdocs	Integer	Maximum number of documents returned
colfields	Array	List of column numbers and or fieldnames to be returned
mvalsep	String	Multi-value separator
returnUNID	Boolean	If true it returns the UNID of the document in addition to the requested fields, otherwise the NoteID
rowformat	Boolean	If true data is returned by row rather than by column (see Return value)

Return value:

Array: Any

Serialization:

Due to data serialization the returned value is a single dimensioned array. The row format parameter determines the result of the serialization. See appendix for more information

dbSendDocument

Web service implementation of the Domino class method NotesDocument.Send().

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
docUNID	String	Universal ID of the document which field values are to be returned
mvalsep	String	Multi-value separator
attachForm	Boolean	Attach form used for the send document (as defined in the Form field)
recipients	Array	List of recipients the document is to be send to

Return value:

String: “true”, “false”

dbSessionKeepAlive

A sort of dummy web service operation that returns the current datetime. This function should be called in a frequency that reflects the Domino server session time out setting.

The Flex/Flash libraries for soapgate Q! call this operation every 10 minutes once the first authentication took place.

dbUploadFile (new)

Allows uploading of files as a Base64 encoded (string) stream into document attachments.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server))
noteID	String	Universal ID or NoteID of the document which contains the attachment
Rtfield	String	Richtext field containing the attachment. If the attachment is not bound to a Richtext field, leave the value blank (“”)
filename	String	Name of attachment (filename excluding filepath, e.g. mypicture.jpg)
replacefile	Boolean	Replace any attachment with the same name
useIBMdecode	Boolean	IBM’s undocumented internal NotesStream.WriteDecoded() method seems to be working correctly. In any case, like for the dbDownloadFile() operation, we rewrote the older base64 decoding LotusScript algorithm in LSXSD.LSS, as it too had issues related to performance. If useIBMdecode is *false*, the performance improved old base64 encoding is used, otherwise the new IBM internal method. **Note: .WriteDecoded() is a multitude times faster than the old LS one. Hence useIBMdecode should always be *true* unless you experience problems.**
base64file	String	XSD_BASE64BINARY stream containing the base64 encoded file

Return value:

String: “1” if the operation was successful, otherwise an error message.

dbUserRoles

Web service implementation of the Lotus Formula function @UserRoles. The function is executed for the authenticated user. To receive the user roles for a specific user or group for a selected database use the web service function dbACL.

Parameters:

srvname	server	String	Name of the Domino server on which the source database resides
dbname	database	String	Pathname of the database (relative to the root data directory of the server)

Return value:

Array: String

dbView2XML (changed)

dbView2XML returns a view's content as XML or alternatively as JSON. *dbView2XML* returns all columns of a view (including hidden, use as color and show as icon columns) and it always adds to extra columns: the first providing each document's unique id and the second the position of each document in the view.

This function works in conjunction with the dbViewColumns web service operation. The latter has to be called first before dbView2XML can be called. The reason being is that dbViewColumn is reading in the view's design into a view profile document.

Non categorised views

The returned XML of a non categorised view is always structured item by item (*flat* XML).

Categorised views

Categorised views are by default returned in an item/child *hierarchical* structure, provided the *categorytagname* parameter has a value (e.g. "category"). If the *categorytagname* is blank, even a categorised view is returned in a *flat* structure.

If a view has multiple categorised columns these will be joined to a single category column. The reason for this is that the development of soapgate Q! was initially focused on providing data access to Flex/Flash applications. The Flex advanced datagrid supports hierarchical data with only one tree column, hence this web service operation joins multiple categorised columns to a single categorised columns. For the same reason *dbViewColumns* returns only the design information for the first categorised column in the view.

Response documents in hierarchy views

If the view has a response document hierarchy the return is always in an item/child *hierarchical* structure.

Use as color columns

Color codes for columns of type *use as color* are not returned as individual column data, but are added as XML attribut value to the respective (following) column values.

Paging

dbView2XML allows paging. There are three parameters provided to control paging: direction, startposition, rows (see table below). With these parameters it is possible to read basically any portion of a view.

Multivalue columns

dbView2XML is currently the only web service operation with a fixed multivalue serialization using the ";" (semicolon).

Note:

- In release 1.0 the documented feature to return categorised views as flat data (instead of hierarchical data) using a blank *cattagname* parameter is now working.
- The new parameter *simple* controls whether additional formatting information available in the view design is used (false) or not (true). Some design information cannot be extracted from the LotusScript NotesView class, namely the view column's Use As Color flag. For this reason dbView2XML is calling an agent to export the view design to DXL, to store it in user profile document (hidden) and then parses the DXL. This process is time consuming and if only the data is desired or if color column coding can be omitted, then the *simple* parameter should be set to *true*.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name of the view which content is to be returned as XML
cattagname	String	Tagname used for the category item. If blank even categorised views are returned as flat XML. This should match the same named parameter of the dbViewColumn operation
catdefault	String	Default string for not categorised items (Notes categorised column returns blank value), e.g. “Not categorised”
direction	Integer	NAVIGATE TO LAST PAGE *= 2* NAVIGATE TO NEXT PAGE *= 1* NAVIGATE TO PREVIOUS PAGE *= -1* NAVIGATE TO FIRST PAGE *= -2* NAVIGATE TO SAME PAGE *= 0* READ DESIGN *= -99*
startpos	String	Position in view from where to navigate into the requested *direction.* If *startposition* is blank it is the first document in the view, otherwise *startposition* must be a valid position of a document in the view; the value that is always returned in the last column.
rows	Integer	Number of rows to be returned. 0 (zero) = all rows
frmtVal	Boolean	If true all column values are formatted to string according to the view column settings, otherwise a simple text conversion takes place.
simple	Boolean	If true the operation will NOT refer to the view’s design for enhanced formatting information, such as columns tick marked as Use As Color. The operation performs much faster as for one the view design has not to be exported to DXL to receive some of the view properties not available through the NotesView class and also because the additional information has not to be added to the output. If false, all programmatically available view design information is returned or used for the data generation/formatting respectively.
returnJSON	Boolean	If true the view content is returned as JSON rather then XML.

Return value:

String XML/JSON

dbViewColumns

dbViewColumns returns the design of a view in XML. In comparison to DXL it is a very much simplified XML. As an alternative the return can be switched to JSON.

Categorised views

If a view has multiple categorised columns these will be joined to a single category column. The reason for this is that the development of soapgate Q! was initially focused on providing data access to Flex/Flash applications. The Flex advanced datagrid supports hierarchical data with only one tree column, hence this web service operation returns the design information for the first categorised column only. For the same reason *dbView2XML* joins the data of multiple categorised columns to a single categorised column.

Use as color columns

The design info of columns of type *use as color* are not returned, but the respective (following) columns (effected by the use as color column) have the tagname *showascolor* set to 1. Which in this context means the column is effected by a *use as color* column. The XML/JSON column data of such columns (returned by *dbView2XML*) have a color attribute set.

Caching

Extracting the view design is a twofold process of which one is to export the view design as DXL and parsing the latter. As exporting a view design as DXL requires designer access level, the soapgate Q! database contains an agent running with the required access level. To avoid this process taking place with every call of dbViewColumns or dbView2XML the DXL is cached in a view specific profile in the soapgate Q! database. The cache expires after 24 hours after which the design is exported to DXL again. During development time the cache parameter allows to read the view design with every call.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name of the view which design is to be returned as XML
cattagname	String	Name overwrite for the first categorised column. This should match the same named parameter of the dbView2XML operation
cache	Boolean	If true the cached design is returned (if available), otherwise the view design is extracted through DXL export and the cache updated
returnJSON	Boolean	If true the view design is returned as JSON rather then XML.

Return value:

String XML/JSON

dbViewFTSearch (changed)

Web service implementation of the Domino class method NotesView.FTSearch() with extended features:

- returns one or more columns
- returns columns and or fields
- always returns the noteID or UNID (last returned column)

Note:
- the parameters *addUNID* and *addNoteID* are now replaced by a single parameter *returnUNID*.

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewname	String	Name or alias of the view, which data is to be returned
query	String	The full-text query. Please refer to the Lotus Notes Help for syntax information
maxdocs	Integer	The maximum number of documents you want returned from the query. Set this parameter to 0 to retrieve all documents (up to 5,000 - see the Usage section for more details) that meet the search criteria
columnfields	String	List of column numbers and or fieldnames in a single string separated with the character passed on in mvalsep
mvalsep	String	Multi-value separator
returnUNID	Boolean	If true it returns the UNID of the document in addition to the requested fields, otherwise the NoteID
rowformat	Boolean	If true data is returned by row rather than by column (see Return value)

Return value:

Array: Any

Serialization:

Due to data serialization the returned value is a single dimensioned array. The row format parameter determines the result of the serialization.

See Appendix for more information

dbViews

Web service implementation of the Domino class property NotesDatabase.Views().

Parameters:

srvname	String	Name of the Domino server on which the source database resides
dbname	String	Pathname of the database (relative to the root data directory of the server)
viewonly	Boolean	If true returns views only
folderonly	Boolean	If true returns folders only
sharedonly	Boolean	If true returns shared views and or folders only
privateonly	Boolean	If true returns private views and or folders only
includehidden	Boolean	If rue returns also hidden views and folders

Return value:

String: XML

The label attribute contains the view name, the data attribute contains the alias name of the view or if not alias is available the view name.

Appendix

Notes field and data types:

1084	ATTACHMENT	
1076	AUTHORS	common or fully qualified names
1024	DATETIMES	date-time value or range of date-time values
1090	EMBEDDEDOBJECT	
21	HTML	HTML source text
25	MIME_PART	
1074	NAMES	common or fully qualified names
7	NOTELINKS	link to a database, view, or document
4	NOTEREFS	reference to the parent document
768	NUMBERS	number or number list
1075	READERS	Common or fully qualified names
1282	RFC822Text	RFC822 Internetmail text
1	RICHTEXT	
8	SIGNATURE	
1280	TEXT	text or textlist

There are more data types. Please refer to the Lotus Notes Designer documentation.

Notes Database Access Levels:

Notes database access levels as returned by the respective ACL related web service functions:

0 = ACLLEVEL_NOACCESS
1 = ACLLEVEL_DEPOSITOR
2 = ACLLEVEL_READER
3 = ACLLEVEL_AUTHOR
4 = ACLLEVEL_EDITOR
5 = ACLLEVEL_DESIGNER
6 = ACLLEVEL_MANAGER

Data Serialization

Domino web services support only single dimensional arrays for operation parameters as well as return values. For this reason soapgate Q! web service operations that do return data of multiple view/folder columns or fields return a serialised single dimensional array.

As a result an array of values collected by e.g. dbcolumnX for the first 3 columns of a view looking like this...

Column1 Row1	Column2 Row1	Column3 Row1
Column1 Row2	Column2 Row2	Column3 Row2
Column1 Row3	Column2 Row3	Column3 Row3

...is converted to a single dimensional array looking like this...

Column1 Row1
Column1 Row2
Column1 Row3
Column2 Row1
Column2 Row2
Column2 Row3
Column3 Row1
Column3 Row2
Column3 Row3

For some of the provided web service operations the parameter *rowformat* offers an alternative way for the data serialization to work:

rowformat= True	rowformat= False
row1_column1 row1_column2 row1_column3 … row2_column1 row2_column2 ...	row1_column1 row2_column1 row3_column1 … row1_column2 row2_column2 ...

Data serialization also affects multi-value column entries or fields. Multi-value column and field values will be converted to type string and each value separated by the character passed on in the *mvalsep* parameter available to all effected web service operations. The data types for single value columns and fields remain intact.

The consumer of the soapgate Q! web service operation must reverse the serialization and create a 2-dimensional array or array collection from the returned single dimensional array. It also needs to revert back serialized multi-values.

The Flex/Flash libraries for soapgate Q! (for Adobe® Flash® Builder™ 4) are doing this for all respective web service operations where the above applies.

Multi-value separator considerations:

The character used in *mvalsep* (default "~" tilde) cannot be part of any of the data columns and or fields received from or send to the server.