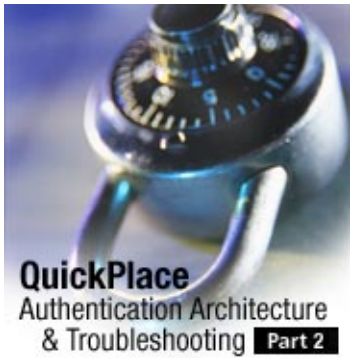
Lotus Developer Domain
The Technical Resource for Lotus software
LDD Today



Level: Intermediate
Works with: QuickPlace
Updated: 01-May-2003

by
Joe
Russo

In **part one** of this series, we looked at a few of the security features supported by QuickPlace, including Basic Authentication, single sign-on, and the QuickPlace security and user directory settings. If you were having authentication problems, maybe you were able to solve those problems using the information in part one. Or perhaps you're still experiencing authentication problems. In this article, we look at some other authentication-related areas, including:

- QuickPlace as both a DSAPI server and application implementation
- User directory customization and configuration
- Authentication, authorization, and LDAP debugging

Like the first article, this one is intended for experienced QuickPlace system administrators. An understanding of Domino administration, LDAP directories, and XML is helpful.

QuickPlace DSAPI

DSAPI is the Domino server API, and you can it use to customize authentication with the Domino Web server. QuickPlace is both a DSAPI server and a DSAPI implementation for the Domino server. We discuss both cases in this article.

QuickPlace DSAPI server

[Note: the following section describes QuickPlace and Netegrity Siteminder integration. See the IBM Lotus Support document "**QuickPlace: How the Netegrity Web Agent and QuickPlace DSAPI interfaces work**" for more information.]

QuickPlace implements the DSAPI server interface, meaning that you can plug a DSAPI application into QuickPlace for use with QuickPlace. To enable QuickPlace to load a DSAPI application, set the following variable in the server Notes.ini:

QuickPlaceDSAPIModules=<*DSAPIDllName*>

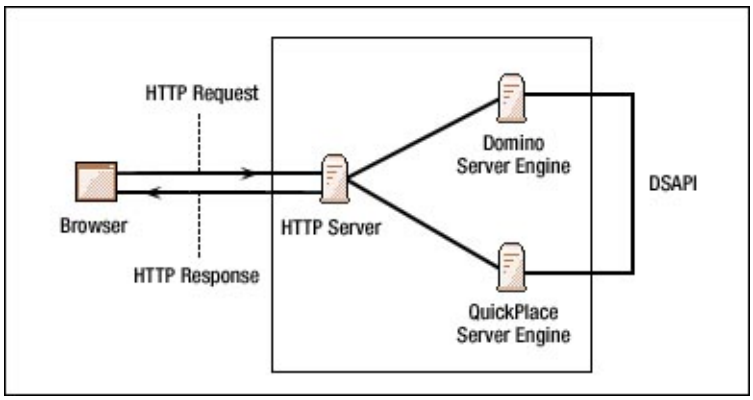
When the QuickPlace server starts up, you should see a notification in the server console informing you that the DSAPI application was successfully loaded by QuickPlace. There is a debug setting for logging DSAPI that you can add to your server Notes.ini file:

QuickPlaceDSAPILogging=5 *(to get the most verbose logging level)*

QuickPlace DSAPI application

To understand why QuickPlace is a DSAPI application, we need to explore how QuickPlace functions as an HTTP server. The following diagram shows the QuickPlace-Domino DSAPI architecture. Note that the box drawn around the HTTP, Domino, and QuickPlace servers represents one Domino server with QuickPlace installed on the same

machine.



A QuickPlace client can issue three kinds of server commands (or requests):

- A request to a QuickPlace specific command
- A request to a QuickPlace command which is not handled by QuickPlace, but by Domino
- A request to a Domino command

QuickPlace determines whether or not to handle a command by looking at the URL request. If the request is a QuickPlace command it uses the following form:

http://servername/quickplace/yourplacename…

If it does not have the /quickplace/ string in this form, then the request is treated as a Domino specific command.

Let's consider what happens to process each one of these kinds of requests.

QuickPlace specific command

When the server is being requested to process a QuickPlace specific command, these steps are taken:

1. The browser issues a request to a QuickPlace specific command.
2. The HTTP server fields this request, but first gives QuickPlace a chance to handle the request.
3. The QuickPlace server examines the request and notes that this is a request from a QuickPlace client.
4. It then inspects the request type and notes that this is a QuickPlace specific command, so it processes the request.
5. As part of handling a request, the user must have access to do so. QuickPlace handles the authentication needed for this request.
6. QuickPlace sets a response for this request and returns control to the HTTP server.
7. The HTTP server sends this response on to the browser.

QuickPlace command that is handled by Domino

To issue a Domino command, the QuickPlace client follows the same steps described in the previous section.

Domino specific command

When the server is being requested to process a Domino specific command, these steps are taken:

1. The browser issues a request to a QuickPlace specific command.
2. The HTTP server fields this request, but first gives QuickPlace a chance to handle the request.
3. The QuickPlace server examines the request and notes that this is a request from a QuickPlace client.
4. It then inspects the request type and notes that this is *not* a QuickPlace specific command, so it returns control to the HTTP server.
5. The HTTP server then gives Domino a chance to handle this request.
6. As part of handling a request, the user must have access to do so. Domino handles the authentication needed for this request. As part of authentication, Domino calls out to the DSAPI application that has registered the authentication and group list events.
7. QuickPlace is a DSAPI application, so it inspects the request and sees that it is from a QuickPlace client. It processes the authentication and returns control to Domino.
8. Domino sets a response for this request and returns control to the HTTP server.
9. The HTTP server sends this response on to the browser.

Note that if Step 7 is not executed, then any Domino specific commands that are part of the QuickPlace client fail authentication or authorization. This is why QuickPlace must be a DSAPI application for Domino.

DSAPI troubleshooting
To troubleshoot these steps you can set the following variables in the server Notes.ini file:

QuickPlaceAuthenticationLogging=5
QuickPlaceDSAPILogging=5

These variables return verbose logging information. The values for these logging parameters are intended to be levels. Each level gives you more verbose logging output. Level = 1 is the lowest level of output. As you increase this value, you get more output—5 being the highest level.

User directory configuration and customization

In release 3.0 of QuickPlace, we introduced a new configuration file, qpconfig.xml, because we wanted QuickPlace to use more open standards. For this release of QuickPlace, we could not move all of the various settings that already existed into the XML file. Our model for this release was to put any new configuration settings added to QuickPlace into this XML file and to migrate the pre-existing ones in a future release.

You can use the qpconfig.xml file to configure LDAP schema mapping and filtering. See the **QuickPlace Administration Guide** for details of the XML configuration file.

QuickPlace access control
QuickPlace makes use of the Domino database access control list (ACL) for managing access. When you log into your Place, there is a list of DNs generated for you called a Names List. This list of DNs is compared to the database's ACL for any matches. An ACL is composed of the DNs of people and groups who have been added to the Place.

For example, let's say we add the user Lee Russo to the Place USASoccer. His DN is:

cn=Lee Russo, ou=United States, o=FIFA

When QuickPlace creates a Place, there are two databases that are created immediately: Main.nsf, which is the main content store for the Place, and Contact1.nsf, which is the membership database. In the ACL for Main.nsf of the USASoccer Place, we add Lee's DN, so the ACL includes:

cn=Lee Russo, ou=United States, o=FIFA

among other names as well. The ACL entry for this DN is associated with some access level and privileges. In QuickPlace, there are three different levels.

Access level	Description
Reader	Allows read access to data in this database.
Author	Allows read and write access to data in this database.
Manager	Includes all the privileges of Author and plus access to the Place administration functions.

Suppose you add Lee as a Reader. In addition to the privileges, there are also Roles defined in the ACL. For QuickPlace, there are three possible roles.

Role	Description
H_Members	Role assigned to all members of the Place
H_Managers	Role assigned to all managers in this room of the Place
H_SuperUser	Role assigned for super user access, which is set by the administrator and is used in all Places on this server

Suppose you add a group to this Place, one which Lee is a member of. The group is MLSPlayers; it's DN is cn=MLSPlayers, o=USSoccer. So now the ACL for Main.nsf looks like this:

cn=Lee Russo,ou=United States, o=FIFA
cn=MLSPlayers, o=USSoccer

You add this group with Author access. One more thing for our example—you create a room in this Place with the name Scoring. QuickPlace creates a new database in this Place to store the content of this room; the database name is of the form PageLibrary*UNID*.nsf (that is, the database's name is PageLibrary and the database's UNID—unqiue id—form the database name). You only add this group to this room, so the room's ACL lists

cn=MLSPlayers, o=USSoccer

only, and this group has Manager access.

When Lee logs into this place, he gets a Names List with his DN and the DN of the MLSPlayers group. When he logs in, he is directed to Main.nsf, where he has Reader access. Note that he has access to this database both *explicitly* (because his personal DN is listed in the ACL) and *implicitly* (because a group which he is a member of is listed in the ACL). He also sees a link on the side bar with the name Scoring, which links to the room. When he clicks this link, he is then directed to this PageLibrary*UNID*.nsf database, where he has Manager access.

Here are the rules for determining access in a database:

- If a user is listed explicitly in an ACL, that is the access level she or he is granted to the database, regardless of any implicit entries in the ACL.
- If a user is listed only implicitly in an ACL, then the access is granted at the *highest* access level granted to the groups in which he/she is a member.

One easy way to inspect an ACL is through the Notes client. Run your Notes client on the QuickPlace server and open Main.nsf or PageLibrary*UNID*.nsf database from a QuickPlace, then select File - Database - Access Control.

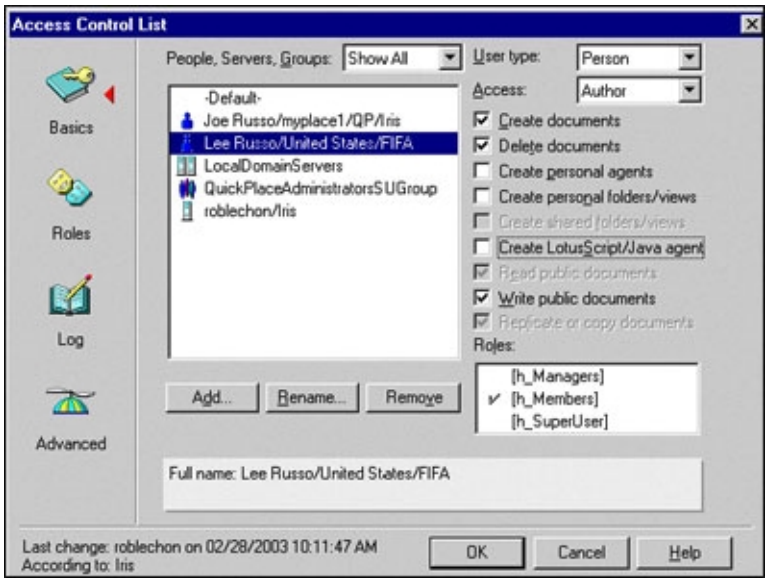
The Access Control List dialog box shows the ACL list and the access rights (in Notes terms) for those entries. Note that the Notes UI represents DNs in the ACL in abbreviated format when possible. Abbreviated format takes the DN and removes the name components—cn, ou, and so on—and equal (=) signs from the DN, separating the values with the forward slash (/) delimiter. So in the UI, our Main.nsf shows the ACL as:

Lee Russo/United States/FIFA
MLSPlayers/USSoccer

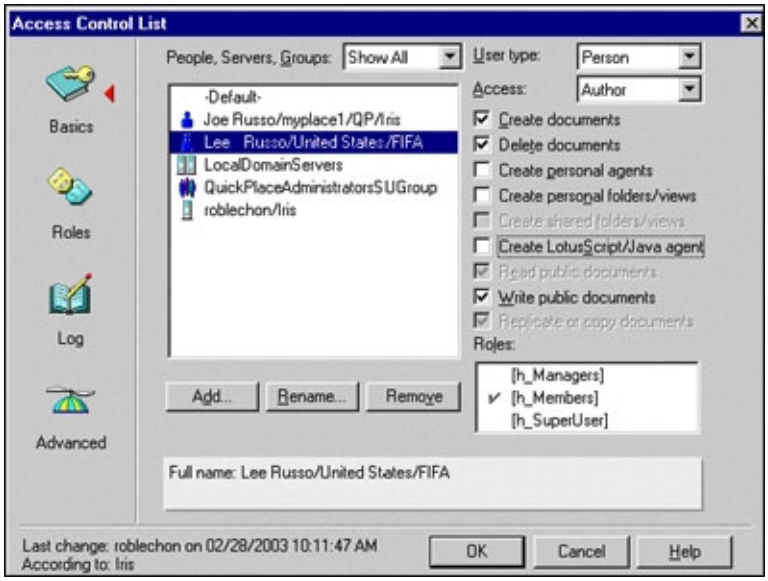
When a user experiences authorization failure, this means that even though he was successful in authenticating, the Names List does not allow him access at any level to the database he is attempting to open.

To troubleshoot this problem, inspect the ACL of the database. Does it list the DN of the user? Or any of his/her groups? Double check the entire DN string. Access is granted by doing a string comparison of the DNs in the user's Names List with those in the ACL. If one of the strings is similar but off by something as trivial as a space character, the string comparison fails. Often authorization failures can be tied to mismatching DNs in the ACL. To determine a user's Names List, refer to the Authentication and Authorization Debugging section that follows.

The following screen shows an ACL with the correct name.



But if the ACL entry for Lee is like the following screen, he is denied access.



Authentication and authorization debugging

QuickPlace has built-in debugging tools that you can turn on to debug authentication and authorization failures. To turn on this debugging, you need to edit your Notes.ini file, located in the program directory of your QuickPlace installation. Edit this file and add the following settings, depending on what you want to debug:

Setting name	Level	Description
QuickPlaceAuthenticationLogging	5	Shows authentication attempts, cache checks, DN of user, and DN for all groups.
QuickPlaceDSAPILogging	5	Shows authentication data passed through the DSAPI interface. This setting is used for DSAPI filters that are interacting with QuickPlace as well as the Domino-QuickPlace DSAPI interactions. Refer to the

		QuickPlace DSAPI section earlier in this article for details on this interaction.
QuickPlaceUserDirectoryLogging	5	Shows the QuickPlace server interactions with the user directory server. It is a great tool to debug the communications with the user directory.

Note that these settings have a level value associated with them. You can set a value for these settings from 0 to 5. A setting of 0 turns off logging. As you increase the levels, you increase the amount of debug information that is output. A level of 1 gives basic information, while a level of 5 gives very verbose logging information.

After you alter your Notes.ini, save it, then stop and restart the server. The QuickPlace server loads these settings at startup and performs the logging. Here are a few sample output sections (note that this sample uses Session Authentication).

The following is directory configuration output:

```
HuDirectoryCustomization::GetCustomizationSettings from xml
HuDirectoryConfig::Copy settings->
type: <LDAP>
name: <mcj.iris.com>
baseDN: <>
groupBaseDN: <>
userName: <>
10:47:16 AM password: <NULL>
useCredentials: <TRUE>
port: <389>
narrowSearches: <FALSE>
ssl: <FALSE>
sslOptions: <11>
sslProtocolVersion: <0>
searchTimeout: <120>
authTimeout: <120>
```

Now a user logs in and the output is:

```
HttpFilterProc - setting httpUser as Christopher Russo
HttpFilterProc - request line = /QuickPlace/quickplace/Main.nsf?Login
HttpFilterProc - URL is Haiku
HttpFilterProc - get session cookie NULL
AuthUser:SetExternalAuthName -
WebUser: SetUser name=Christopher Russo
WebUser: SetHaikuName as = Christopher Russo/quickplace
HttpFilterProc - Authenticate Event
WebUser: SetUser name=Christopher Russo
WebUser: SetHaikuName as = Christopher Russo/quickplace
AuthUser: Authenticate - checking cache
AuthUser:CheckCache for : Christopher Russo/quickplace is not found in cache
```

QuickPlace authenticates the user and the output shows:

```
AuthUser: LookupPassword URL is QuickPlace - let QuickPlace authenticate for Christopher Russo/quickplace
HuContactsDb::GetAuthenticName for : Christopher Russo
HuContactsDb::GetAuthenticName for : Christopher Russo is a common name
HuContactsDb::GetAuthenticName for : Christopher Russo common name 1 matches found for 1 NoteIds
HuDirectoryBaseClass::AuthenticateAndAuthorize calling Authenticate for CN=Christopher
Russo/O=NERevolution
HuDirectoryBaseClass::Authenticate attempt for : CN=Christopher Russo/O=NERevolution against directory
mcj.iris.com
HuDirectoryBaseClass::GetLdapSessionHandle via Ldap unsecure to contact mcj.iris.com
HuDirectoryBaseClass::Bind as CN=Christopher Russo,O=NERevolution = FALSE for ldap_version = 1
HuDirectoryBaseClass:: Bind for handle=B0160DC
HuDirectoryBaseClass::Authenticate CN=Christopher Russo/O=NERevolution is not verified against directory
```



```
mcj.iris.com
HuDirectoryBaseClass::AuthenticateAndAuthorize returned from Authenticate
HuContactsDb::GetAuthenticName for : Christopher Russo is NULL
```

Authentication failed because the user entered the wrong password, so he tries again with the correct one. Here is the output:

```
HuDirectoryBaseClass::GetLdapSessionHandle via Ldap unsecure to contact mcj.iris.com
HuDirectoryBaseClass::Bind as CN=Christopher Russo,O=NERevolution = TRUE for ldap_version = 3
HuDirectoryBaseClass:: Bind for handle=B0280DC
HuDirectoryBaseClass::Authenticate CN=Christopher Russo/O=NERevolution is verified against directory
mcj.iris.com
```

Now, QuickPlace gets his groups. The output looks like the following:

```
HuDirectoryBaseClass::AuthenticateAndAuthorize returned from Authenticate
HuDirectoryBaseClass::AuthenticateAndAuthorize calling Authorize
HuDirectoryBaseClass::GetGroupsForName for : CN=Christopher Russo,O=NERevolution NestDepth=1
NestLimit=1
HuDirectoryBaseClass::Authorize time=491 for CN=Christopher Russo/O=NERevolution
HuDirectoryBaseClass::SetUserGroups for : CN=Christopher Russo/O=NERevolution
HuDirectoryBaseClass::SetUserGroups group CN=Soccer Stars/O=International for CN=Christopher
Russo/O=NERevolution
HuDirectoryBaseClass::GetContactInfo for: CN=Christopher Russo,O=NERevolution
HuDirectoryBaseClass::DecomposeDN: pDecomposedDN[0]=[CN=Christopher Russo]
HuDirectoryBaseClass::DecomposeDN: index=1 pDecomposedDN[index]=[O=NERevolution]
HuDirectoryBaseClass::DecomposeDN: index=2 pDecomposedDN[index]=[]
HuDirectoryBaseClass:: UnBind for handle=0
HuContactsDb::GetAuthenticName for : Christopher Russo is CN=Christopher Russo/O=NERevolution
HuContactsDb::GetAuthenticName - GetUserGroupNames for CN=Christopher Russo/O=NERevolution
WebUser: SetUser name=CN=Christopher Russo/O=NERevolution
WebUser: SetHaikuName as = CN=Christopher Russo/O=NERevolution/quickplace
WebUser: SetHaikuName as = CN=Christopher Russo/O=NERevolution/quickplace
AuthUser: UpdateCache - User CN=Christopher Russo/O=NERevolution/quickplace added to the cache
```

QuickPlace sets the names list for our user as shown in this output:

```
HttpFilterProc:Authenticate:SetAuthenticUserName - pAuthData->authName = CN=Bob
Marley/O=NERevolution
HttpFilterProc - setting httpUser as Christopher Russo
HttpFilterProc - request line = /QuickPlace/quickplace/Main.nsf?Login
HttpFilterProc - URL is Haiku
HttpFilterProc - get session cookie NULL
AuthUser:SetExternalAuthName -
WebUser: SetUser name=Christopher Russo
WebUser: SetHaikuName as = Christopher Russo/quickplace
HttpFilterProc - UserNameList Event
```

Now, QuickPlace checks the user cache for the next request and uses it!

```
AuthUser: Authenticate - checking cache
AuthUser:CheckCache for : Christopher Russo/quickplace is found in cache
AuthUser: LookupPassword User Christopher Russo/quickplace is found in cache
HttpFilterProc:UserGroupList:IsSameUser - pUserName = CN=Christopher Russo/O=NERevolution and
pUserInfo->GetUserName() = CN=Christopher Russo/O=NERevolution
HttpFilterProc:WriteNamesList: inputs pContext=NOT NULL pUsernameList=NOT NULL pUserInfo=NOT
NULL userGroups=NULL
HttpFilterProc:UpdateGroupList: namesList=NOT NULL
HttpFilterProc:UpdateGroupList: groupName = CN=Christopher Russo/O=NERevolution
HttpFilterProc:UpdateGroupList: groupName = *
HttpFilterProc:UpdateGroupList: groupName = */OU=quickplace/OU=QP/O=Iris
HttpFilterProc:UpdateGroupList: groupName = CN=h_Members/OU=quickplace/OU=QP/O=Iris
HttpFilterProc:UpdateGroupList: groupName = CN=Soccer Stars/O=International
```


HttpFilterProc - setting httpUser as CN=Christopher Russo/O=NERevolution

LDAP debugging

One of the biggest sources of authentication issues with the user directory is the user directory not working properly. You can and should verify that your user directory is functioning normally so that your QuickPlace server runs correctly.

The tool, ldapsearch.exe, is installed when you install your Domino server. In the program directory, find the tool ldapsearch.exe. Run this tool from the DOS prompt, and you'll see the usage information. Here are steps to debugging you LDAP situation:

1. Before you use ldapsearch.exe, ping your LDAP server from your command line, using the following command syntax, replacing yourserverDNSName with the name of your domain name server:

   C:\Lotus\Domino\ping *yourserverDNSName*

   If this does not produce a result, then your Domino server cannot see the LDAP Server. Check with your network administrator to ensure you can connect. Also, check the DNS name for your LDAP server.

2. After successfully pinging your LDAP Server, try to access it through the LDAP protocol. You can perform a simple search to see what happens using the ldapsearch tool. Enter the following command, replacing yourserverDNSName with the name of your domain name server:

   C:\Lotus\Domino\ldapsearch –h *yourserverDNSName* cn=*

   This searches the LDAP server, yourserverDNSName, for all entries that have a cn attribute—which may be a lot. You could try a smaller scale search if you know someone's name in the directory, for instance, from our example:

   C:\Lotus\Domino\ldapsearch –h *yourserverDNSName* "cn=Lee Russo"

3. Now, if this search produces a result, then you know you've got anonymous access to your LDAP server. If this fails, you receive an LDAP error message. Use this information to fix the connection—until you do, QuickPlace cannot access this LDAP server either.

After you've passed the anonymous search, you may want to try an authenticated search (or your directory may not allow anonymous searches). You can use the same tool to debug this connection as well. Suppose that our friend Lee wants to use his credentials to do the authenticated search. Enter the following in the command line:

C:\Lotus\Domino\ldapsearch –h *yourserverDNSName* –D "cn=Lee Russo,ou=United States, o=FIFA" –w iluvsoccer "cn=Lee Russo"

This performs the LDAP search by first binding to the LDAP server as Lee with his password. If the binding (or authentication) is successful, then the search is performed. If this test passes, then you know that you can perform authenticated searches on your LDAP server. You can configure QuickPlace to do authenticated searches (with the same DN and password from the test), so it can perform authenticated searches as well. If this fails, then you should use the LDAP error information to remedy the issue.

You can get usage information to learn more about this utility by typing the following on your DOS prompt:

C:\Lotus\Domino\ldapsearch

Conclusion

The article walked you through the many variations and details of QuickPlace authentication. Please refer to this article whenever you need a handy guide. And most importantly, whenever you encounter these kinds of problems, keep detailed notes on settings, log output, and the scenario causing the problem. Armed with that information and this document, you should be able to clear up most problems.

ABOUT THE AUTHOR

Joe Russo is the Administration Audience lead developer for QuickPlace 3.0. He has a seldom used Electrical Engineering degree from SUNY at Buffalo where he graduated too many years ago to mention. He spent the early part of his career in the imaging and graphics world and then hopped onto the Internet development bandwagon. He lives with his wife and three sons somewhere in Massachusetts, in seclusion from his admiring fans. He hopes one day to write a book that people (other than

friends and relations) would be willing to purchase.

