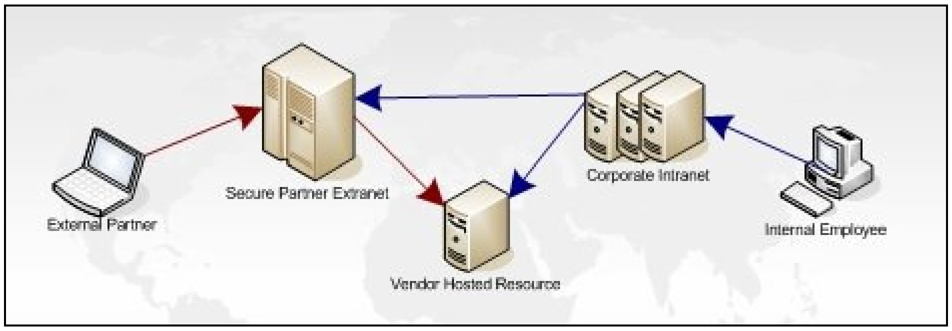
NCT SIMPLE SIGNON

INTEGRATION WITHOUT AGGRAVATION



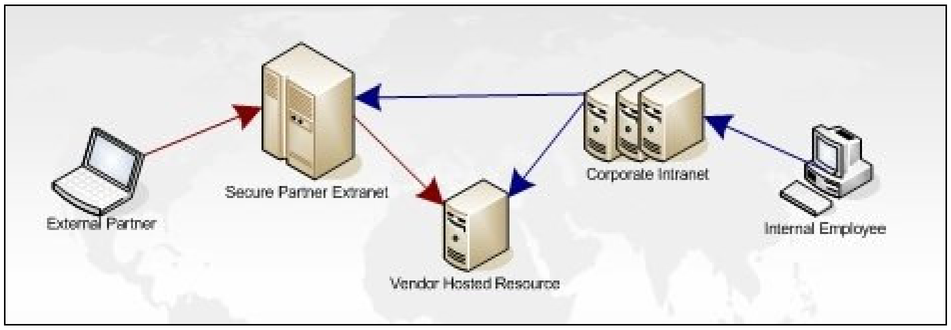
NORTHERN Collaborative TECHNOLOGIES

TABLE OF CONTENTS

INTRODUCING INTEGRATION WITHOUT AGGRAVATION

NCT Simple Signon provides a simple way to define individual shared access relationships on a one to one basis with a variety of different remote sites. Better still, it lets you get there quickly by providing a complete **vender neutral** built in Authentication Schema which is fully documented and ready to share with vendors and remote sites. On the other hand, perhaps someone else has already defined a method for passing user data to you and you need to accept that data and log them in. NCT Simple Signon provides a simple function to do just that.

The whole idea of a network is to connect information and capability from more than one place. Those places are often very different. They use different software, different authentication methods, and are frequently managed by different people.



NCT Simple Signon is a **vender neutral** tool which lets you tie IBM Lotus Domino™ based sites to corporate intranets, applications outsourced to vendors, channel partner networks, and third party applications by providing a seamless mechanism for accepting user credentials without requiring yet another login.

NCT Simple Signon does not force your vendors, partners, or other corporate sites to agree on a single universal sign on solution, and unlike traditional "Single Sign On" tools, NCT Simple Signon does not require you to relinquish all control of your server to some central authentication point – though you can if you like. NCT Simple Signon doesn't even require that it is the only method for authentication on your server, or that you use it in only one way.

NCT Simple Signon provides an easy to manage, secure and reliable way to accept credentials from nearly any other application, and to pass credentials for users authenticated by your server to other sites.

HOW IS NCT SIMPLE SIGNON DIFFERENT?

There is no shortage of tools on the market designed to provide so-called "Universal Login" or "Single Sign-On". What nearly all of these tools assume however, is that a single defined platform, directory, and definition can be universally adopted across all the parts of the system to provide seamless integration. NCT Simple Signon makes no such assumptions.

Suppose your secure partner channel website has an established directory and password system with a highly granular level of access controls. Now suppose you want to provide additional features for some or all of those partners, but those features are to be provided by a third party vendor with whom you have contracted. You may not be the only customer of this vendor, they may not be your only vendor, or they may have an offering incompatible with your naming of security choice.

NCT Simple Signon provides a simple way to define individual shared access relationships on a one to one basis with these kinds of outside managed sites.

WHAT? NO CENTRAL DIRECTORY?

That's right. NCT Simple Signon does not require you to share a directory with all the sites you do business with. By supporting defined name translation, we allow each site participating in a shared authentication schema to maintain its own directories as they best suit the purpose of that site. Nothing prevents you from sharing directory or group information from site to site, we simply do not require it.

WHY DOESN'T NCT SIMPLE SIGNON USE A DSAPI FILTER?

DSAPI is the defined method for making authentication and access changes to the default way Lotus Domino authenticates users. The problem with using DSAPI is that it is in use and in-line as part of every single http transaction that occurs on your server. As a result, even the slightest problem with the way a DSAPI filter is designed or implemented can very quickly produce a cumulative impact sufficient to cause major problems for your server. There are some excellent products on the market making using of DSAPI technology for a variety of things, and if done right it can be an excellent tool. At Northern Collaborative Technologies, we have simply chosen a less problematic approach for NCT Simple Signon.

Unlike a DSAPI filter, NCT Simple Signon makes only a very limited number of calls to the Lotus Domino CAPI. Those calls are made only when specifically needed to authenticate a user, and only once per session for that user. The result is thousands or even tens of thousands of fewer calls to the API, and a drastically reduced likelihood of problems.

SECURITY CONSIDERATIONS WITH NCT SIMPLE SIGNON

At Northern Collaborative Technologies, we've worked hard to make NCT Simple Signon a robust, stable, and secure product for your environment. With any authentication related product, however, there are dangers.

THERE IS NO SUCH THING AS PERFECT SECURITY

If you absolutely, positively, never, ever can allow something to be shared; do not put it on a web site or a server. There are techniques which can be used to watermark items individually and thus trace leaked documentation, but even these are only good at cleaning up after the fact. This tool, like most others, is designed to provide a reasonable level of security to prevent the casual or even modest attempts to circumvent authorized use procedures. If NCT Simple Signon is used correctly, it provides one part of a security solution – but only one part. Application design best practices are still required, as is the localized encryption and other security measures required in modern applications.

Authentication is not the same thing as Authorization. NCT Simple Signon is designed to enable you to pass authentication credentials among disparate sources and maintain a reasonable level of surety that those credentials are authentic. Once you have authenticated a user, however, you must still control that user's authorization to view, add, create, change, or delete data.

NCT Simple Signon is powerful medicine. In fact, more than 90% of the development effort associated with this product has gone into providing the necessary controls to allow its safe use.

USING THE BUILT IN SCHEMA

NCT Single Signon is not just a tool; it's a ready built solution. Northern Collaborative Technologies has included with NCT Simple Signon a complete Authentication Schema which makes use of a widely available cross-platform industry standard encryption algorithm.

The Authentication Schema provided with NCT Simple Signon is designed to be easily shared with your vendors and remote site partners to allow rapid integration without compromising security unnecessarily.

Key features of the built in Authentication Schema

- ☒ Uses the Blowfish encryption algorithm which is readily available as source or in pre-packaged libraries for nearly all development platforms at little or no cost.
- ☒ Includes documentation sufficient for remote site integration – you do not have to write and support your own.
- ☒ Includes built in support for user-name translation, both inbound and outbound, for users with different names on different systems.
- ☒ Includes built in support for limiting specifically which users can use the schema on a per-site basis – preventing users from spoofing an administration login.
- ☒ Includes a pre-built test form you can use to allow remote site vendors to validate their use of the Blowfish encryption algorithms and compare them with yours for compatibility
- ☒ Includes a pre-built test packet generating tool, allowing you to create sample authentication packets for vendors and remote site operators to utilize as they prepare their side of the integration.
- ☒ Includes techniques to prevent the re-use of packets by users either through bookmarks, saved text, or forwarded links.
- ☒ Includes built in support for storing an encryption key for each remote site without exposing that key in a configuration document or source code.

USING THE API

You've read all the nifty things about the built in Authentication Schema but you really want to make your own. Maybe someone else has already defined a schema and you want to take part. In either case, NCT Simple Signon provides an extremely simple mechanism for taking nearly any username and generating a session token for that name. Be careful, however. Like any powerful tool there is plenty of opportunity for misuse. We recommend strongly that you become familiar with the built in schema first, and understand the steps we have taken to ensure safety before you attempt to create your own.

SYSTEM REQUIREMENTS

NCT Simple Signon requires a Lotus Domino version 6 or higher server. NCT Simple Signon has been tested and shown to work properly on IBM Lotus Domino version 7 servers as well.

Although the schema itself is not application server, operating system, or machine specific; if you wish to use NCT Simple Signon for inbound user access you will need at least one Win32 based IBM Lotus Domino server. Future versions will support other CAPI compatible versions of Lotus Domino for this purpose. For use with servers other than Win32 based, you can set up a single Win32 server to generate session tokens in a Domino Shared Session environment.

NCT Simple Signon is not browser specific, and should work for users on nearly any modern web browser including both Microsoft Internet Explorer and Mozilla Firefox.

Administrators of NCT Simple Signon should have an IBM Lotus Notes Client version 6.0 or higher, and must have sufficient rights to install databases to the server.

To function properly, the NCT Simple Signon Application Database will have to be "Signed" with an ID file having sufficient rights to run Unrestricted LotusScript agents.

INSTALLATION

READ THE FRIENDLY MANUAL, PLEASE

NCT Simple Signon is an authentication tool. If used or deployed improperly it can allow access to your system in ways or by users you do not intend. We strongly recommend you read this entire document before deploying NCT Simple Signon to a production server.

OBTAINING A LICENSE KEY

NCT Simple Signon will not operate without a license key specifically tailored for your environment. Limited duration trial keys are available through our web site at http://www.thenorth.com. License keys may be purchased direction from Northern Collaborative Technologies with a variety of prices and limitations. Please use our website to obtain the most current pricing available.

DEPLOYING THE LICENSE KEY AND PROGRAM FILE

NCT Simple Signon is distributed in three primary parts. A license key which is sent separately to you, either through our trial registration site or through other means, a program file in the form of a Win32 Dynamically Linked Library (DLL), and an Application Database.

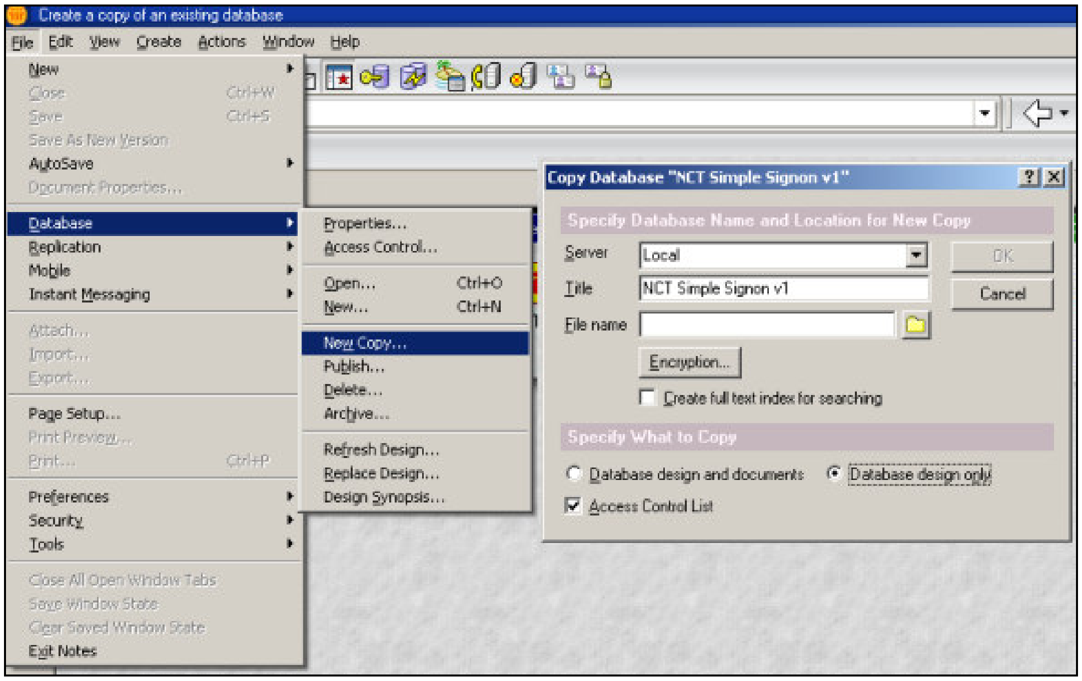
The supplied DLL file and License Key file must be placed in the server's Program Directory. This is the location specified by the "NotesProgram=" parameter in the server's NOTES.INI file. If both the DLL file and License Key file are not present in this location, NCT Simple Signon will not operate correctly.

DEPLOYING THE APPLICATION DATABASE

To prevent misuse, the NCT Simple Signon Application Database must be deployed according to the instructions below.

CREATE A NEW COPY FOR DEPLOYMENT

Make sure to use the "File-Database-New Copy" to create your copy of the Application Database to prevent unwanted replication with other versions or servers.



Copy Database "NCT Simple Signon v1"

Specify Database Name and Location for New Copy

Server: Local

Title: NCT Simple Signon v1

File name:

☐ Create full text index for searching

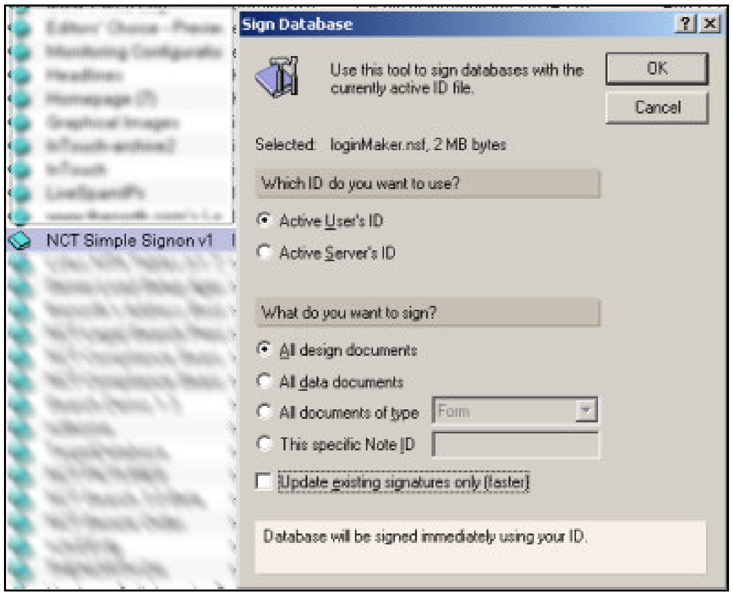
Specify What to Copy

○ Database design and documents [✓ ● Database design only]

☑ Access Control List

SIGN THE DATABASE

Sign the new copy of the database using the IBM Domino Notes Administration Client. You will need to use an ID file for the signature which has sufficient access rights to run "Unrestricted LotusScript Agents".



Use this tool to sign databases with the currently active ID file.

Selected: loginMaker.nsf, 2 MB bytes

Which ID do you want to use?

[✓ Active User's ID]
Active Server's ID

What do you want to sign?

[✓ All design documents]
All data documents
All documents of type Form
This specific Note ID

Update existing signatures only (faster)

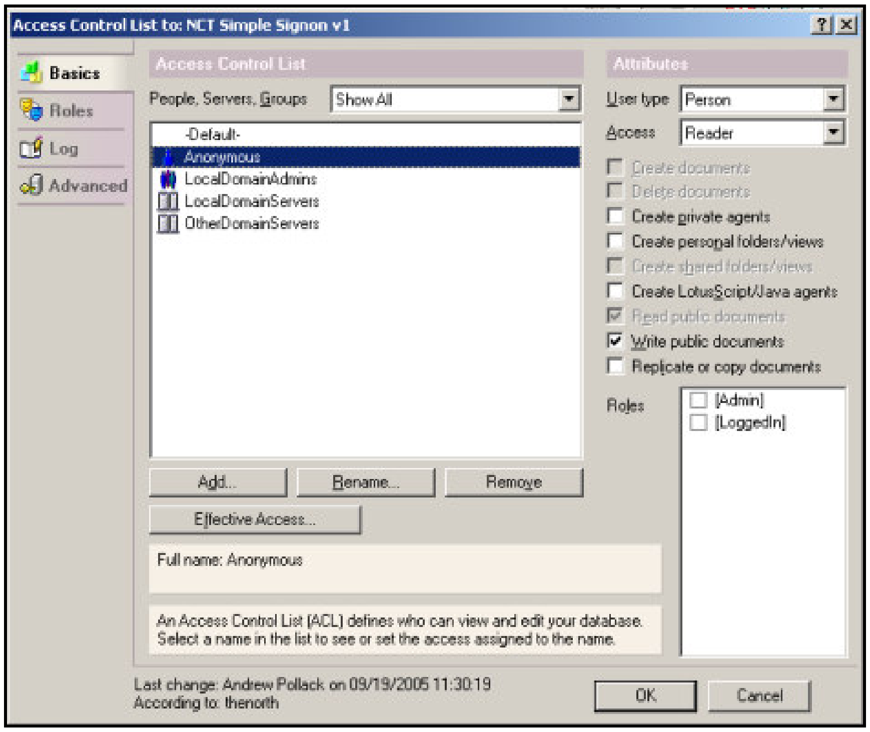
Database will be signed immediately using your ID.

Note: The authority to run Unrestricted LotusScript Agents is an extremely high level of access. A best practice would be to create one or more special ID files used to sign application which must have this level of access and use those for signatures only, rather than allowing specific users this level of access.

CHECK THE DATABASE ACCESS CONTROL LIST

NCT Simple Signon should be configured with the following access control entries:

Identifier	**Access Level**	**Roles Selected**
- Default -	Reader (Write Public Documents)	[LoggedIn]
Anonymous	Reader (Write Public Documents)	- None -
LocalDomainAdmins (or other Administrators)	Manager	[Admin] [LoggedIn]
LocalDomainServers	Manager	[LoggedIn]


Access Control List to: NCT Simple Signon v1

Access Control List

People, Servers, Groups: Show All

- -Default-
- Anonymous
- LocalDomainAdmins
- LocalDomainServers
- OtherDomainServers

Attributes

User type: Person

Access: Reader

- [ ] Create documents
- [ ] Delete documents
- [ ] Create private agents
- [ ] Create personal folders/views
- [ ] Create shared folders/views
- [ ] Create LotusScript/Java agents
- [✓] Read public documents
- [✓] Write public documents
- [ ] Replicate or copy documents

Roles:
- [ ] [Admin]
- [ ] [LoggedIn]

Full name: Anonymous

An Access Control List (ACL) defines who can view and edit your database. Select a name in the list to see or set the access assigned to the name.

Last change: Andrew Pollack on 09/19/2005 11:30:19
According to: thenorth

DEPLOY THE DATABASE TO YOUR SERVER

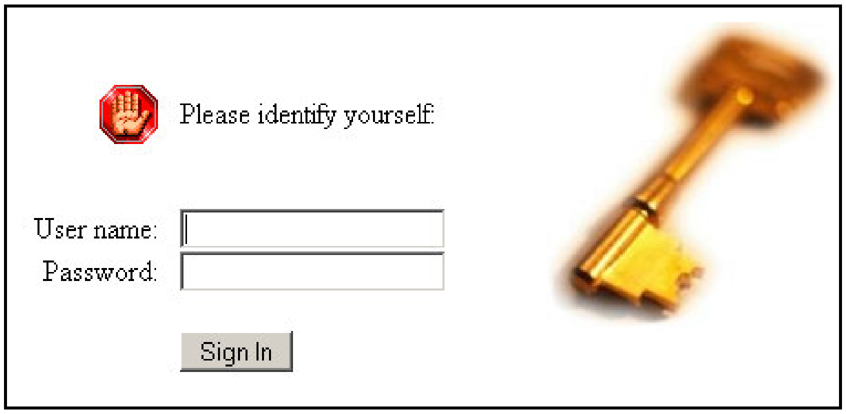
Once all of the steps above have been completed, copy the Application Database to the server. The Application Database can have any filename and live in any directory or folder however we recommend it be placed near the root, and named with a filename that does not contain spaces.

CHECK YOUR DOMINO SESSION BASED LOGIN SETTINGS

NCT Simple Signon works by issuing each user as needed an LTPA Token as if they had accessed the server using the standard session based login form. Before attempting to configure NCT Simple Signon, please make certain that your server is properly configured for Session Based Login.

- ☒ Your Server's "Server" document should be set to "Load Internet configurations from Server\Internet Sites documents"
- ☒ You must have created a "Web SSO Document" in the Domino Directory
- ☒ The Web Site document in your Internet Sites view governing this server should be set to use "Session Based Authentication" of the type "Multiple Servers (SSO)".

To verify your Session Based Login is set correctly, you should see a screen like this when attempting to access a secure database on your server through a web browser:

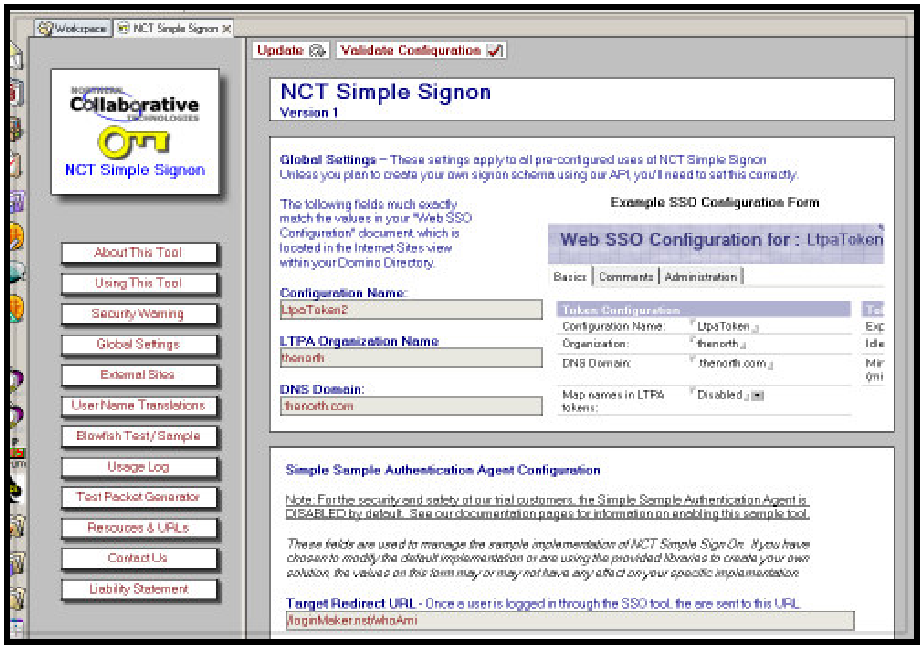

Please identify yourself.

User name:

Password:

SET THE GLOBAL OPTIONS

Once your Application Database has been deployed and you have verified that your Lotus Domino sever is set to handle session based login; open the database and select "Global Settings" to configure your application's defaults.



NCT Simple Signon
Version 1

Global Settings – These settings apply to all pre-configured uses of NCT Simple Signon
Unless you plan to create your own signon schema using our API, you'll need to set this correctly.

The following fields much exactly match the values in your "Web SSO Configuration" document which is located in the Internet Sites view within your Domino Directory.

Configuration Name: LtpaToken2

LTPA Organization Name thenorth

DNS Domain: thenorth.com

Example SSO Configuration Form

Web SSO Configuration for : LtpaToken

Basics | Comments | Administration

Token Configuration	
Configuration Name:	LtpaToken
Organization:	thenorth
DNS Domain:	.thenorth.com
Map names in LTPA tokens:	Disabled

Simple Sample Authentication Agent Configuration

Note: For the security and safety of our trial customers, the Simple Sample Authentication Agent is DISABLED by default. See our documentation pages for information on enabling this sample tool.

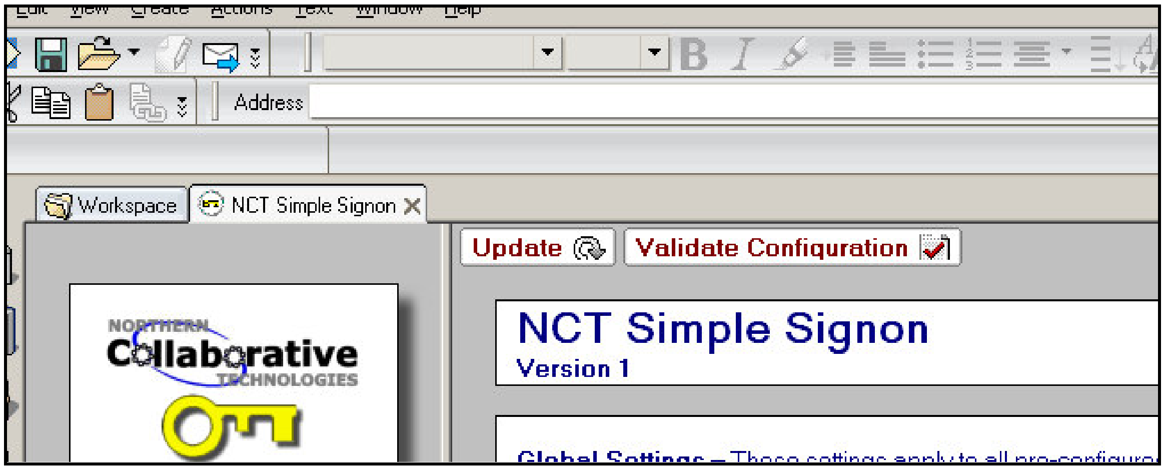
These fields are used to manage the sample implementation of NCT Simple Sign On. If you have chosen to modify the default implementation or are using the provided libraries to create your own solution, the values on this form may or may not have any affect on your specific implementation.

Target Redirect URL - Once a user is logged in through the SSO tool, the are sent to this URL

/loginMaker.nsf/whoAmi

VALIDATE YOUR CONFIGURATION

Use the action button at the top of the Global Options screen to validate your settings. This will cause an agent at the server to access the specified SSO Configuration document and will check for a valid License Key file.



PLAYING WELL WITH OTHERS

There can be no valid point to an authentication sharing tool if it doesn't play well in the sandbox with others. To do that, you need an agreed way to pass user credentials around.

NCT Simple Signon is designed to be out of the box ready to play in that game. We've included documentation and tools specifically designed to be easy for other vendors to work with. On average, we've found less than five working days are required for most sites to implement their side our defined schema, and many can do it in just a few hours.

DEFINING A LOGIN SCHEMA

WHAT IS A LOGIN SCHEMA, ANYWAY?

A "Schema" is just a specification for how things are to be done. In this case, for how user credentials should be passed from one source to another in a secure manner.

SCHEMA SECURITY CONSIDERATIONS

If you're planning to develop a schema of your own; or if you're evaluating someone else's, you'll need to consider a few key things. <u>Here's the short list we came up with:</u>

Simplicity

The complex the solution is to implement, the more likely it will meet resistance or be done incorrectly. This can be a difficult trade-off point when selecting encryption and hashing tools.

Interoperability

The whole point of passing credentials becomes null and void if you require parts which are not easily available on a very wide variety of platforms.

Security vs. Obscurity

Just because something is hidden, does not mean it is secure. There are people out there with nothing better to do than to figure out what you've done and try to find ways around it. Real security requires validated, published, peer-reviewed encryption tools.

Prevent Token Spoofing

An authentication token that can be copied and re-used, or can be re-created without some secret key can be easily spoofed. A shared secret key is one of the most common methods for passing data which cannot be read.

Prevent Token Saving and Sharing

Just because you can't read something, doesn't mean you cannot copy it or re-use it. Techniques that include time stamps and make encrypted packets useless after a predetermined time limit prevent users from simply creating bookmarks with their encrypted strings on the URL.

Control Access to the Encryption Key

Once you've agreed on an encryption key with a remote site, where you store that key is critical to your security. If you hardcode it in your program code, or store it in a document where it can be easily read, you may as well give it away.

USING THE BUILT IN SCHEMA

Based on the short list you see above, Northern Collaborative Technologies has created a schema and provided a full implementation of that schema to get your off to a fast start. It is by no means perfect, but it is based on techniques and implementations which have been in continuous use for several years by individual clients and has proven reasonably easy for remote site operators to understand and join.

OVERVIEW OF THE BUILT IN SCHEMA

The built in Authentication Schema really comes down to just a few key parts. They are an agreed encryption algorithm, a shared key, and a defined packet of data which includes the user credentials and a time stamp. The user credentials are attached to the timestamp and the whole packet is encrypted and sent to the remote site on the URL.

When the remote site receives the data, by decrypting the packet it can be certain that the packet was either created by the site authorized to do so, or by someone else who had been given that site's encryption key. Taking the next step, by validating that the time stamp has not expired, a site can be certain that the packet was created within a short enough time window to prevent saving or sharing the packet itself. The site can thus be certain that the user came from a specific remote site, recently, and that the remote site attests to that user's credentials.

THE SCHEMA DEFINITION

This section defines the actual packet passed between the Source site and the Remote site. The packet itself is passed on the URL as part of a parameter as defined later in this document.

The Transfer URL

The NCT Simple Signon Application Database is designed to accept users with the following url:

http://serverfqdn/applicationDb/NCTSchemaUserAuth?OpenAgent&ref=XXXX&pkt=YYYY

In this case, XXXX is the reference id for the external site configuration document, and YYYY is a packet of encrypted data containing the user name and a time stamp.

For outbound users, the remote site must provide a transfer URL using some similar form. The site reference configuration document within the NCT Simple Signon Application Database provides a place for this to be entered, where the string %%% will be substituted for the packet itself. This URL is not the one presented to users of the site. Users see an internal which looks like this:

http://serverfqdn/applicationDb/NCTSchemaUserOut?OpenAgent& ref=XXXX

When users link to this URL, the site reference represented by XXXX is used to create an outbound packet and redirect the user to the URL provide. For example, if a remote site specified that users transferred to them were to use the url:

http://remotesite/cgi-bin/LoginUser.cgi?userdata=%%%

That would become the URL users of this site would be redirected to after the packet was created.

The Encrypted Packet Itself

The packet itself is defined as a sequence of data strings, as follows:

[NN] [User Info] [YYYY] [Mo] [Da] [Hr] [Mi] [Se]

For example:

25JoeUser20050918153022

The first two digit numeric value is then added to each of the six time date values at the end. This prevents an obvious pattern in the encrypted data when using an ECB mode encryption schema. The resulting string now looks like this:

25JoeUser20303443405547

Finally, the packet is encrypted using the Blowfish algorithm and the agreed on encryption key, resulting in a string like this (the encryption key "password" was used in this case):

F9512613FFBA00E2986215B2BB6D2315DED7BF53C8FF2C97

Each byte in the encrypted string is represented as a pair of hexadecimal digits. The encrypted string will always be an even number of digits in length.

The Data Packet – Prior to Encryption

Characters	Value
00-01	A random number between 00 and 99, unique for each packet created. Each numeric value in the date stamp is increased by this number prior to encryption. This prevents a visible pattern from showing up in data encrypted with the Blowfish encryption algorithm in ECB mode.
02-xx	The user credentials. Usually this is the user name; however it can also include other data in some delimited format. This is the "payload" itself. Whatever is being transferred from one system to the other is located here. It takes the full length of the packet except the final fourteen characters which represent the date stamp.
End-10 through End-13	The current year at the time the packet was created, plus the numeric value from first pair of digits in the packet. Always in GMT.
End-08, End-09	The current month at the time the packet was created, plus the numeric value from the first pair of digits in the packet. Always in GMT.
End-06, End-07	The current day at the time the packet was created, plus the numeric value from the first pair of digits in the packet. Always in GMT.
End-04, End-05	The current hour at the time the packet was created, plus the numeric value from the first pair of digits in the packet. Always in GMT.
End-02, End-03	The current minute at the time the packet was created, plus the numeric value from the first pair of digits in the packet. Always in GMT.
End-01, End	The current second at the time the packet was created, plus the numeric value from the first pair of digits in the packet. Always in GMT.

Packet Encoding

Encrypting data yields a result in bytes which may not be valid for transfer over a URL reference. For this reason, each byte in the encrypted string is represented as a pair of hexadecimal digits. This is to say, a byte with the value of 255 would be represented as FF, while a byte with the value of 10 would be represented as 0A. The result will be an even number of characters either numbered between 0 and 9, or letters between A and F. The string is not case sensitive.

The Blowfish Encryption Algorithm

One of the most critical choices to make when creating an Authentication Schema is which encryption algorithm to use. It may seem illogical at first, but one of the most important aspects of encryption is that the algorithm is publicly available and subject to a peer review in that community. Any encryption which relies on its algorithm rather than its key being secret is not very secure at all. For our use, we've settled on "Blowfish".

What is Blowfish?

From Blowfish creator Bruce Schneir's web site:

"Blowfish is a symmetric block cipher that can be used as a drop-in replacement for DES or IDEA. It takes a variable-length key, from 32 bits to 448 bits, making it ideal for both domestic

and exportable use. Blowfish was designed in 1993 by Bruce Schneier as a fast, free alternative to existing encryption algorithms. Since then it has been analyzed considerably, and it is slowly gaining acceptance as a strong encryption algorithm. Blowfish is unpatented and license-free, and is available free for all uses."

You can find out more specific details about Blowfish, obtain source code to create your own implementation, and find links to free and commercial implementations of Blowfish for most programming languages at this site. http://www.schneier.com/blowfish.html

Why did we choose Blowfish?

Blowfish represents an easy to implement, cost effective solution. Implementations are so widely available that it makes an excellent cross platform choice. When sending a specification to remote sites, the use of Blowfish ensures that those sites will be able to participate with minimum cost and difficulty. Blowfish has been heavily tested and analyzed by top security experts, and its level of safety is well known.

What Blowfish Options Are We Using?

Like many encryption algorithms, Blowfish has been implemented in a variety of ways. These break down into two primary types. The first, ECB – short for "Electronic Code Book" is the simpler of the two. In "ECB" mode, each block of eight bytes is encrypted using the algorithm as a block, and the blocks are simply concatenated to make a longer string. If you have text which is 32 bytes long, it will be processed as a series of four independent blocks. The other modes of operation make use in one way or another of modifying each block based on the previous block. For example, in Cipher-block chaining (CBC) mode, each block of data is modified with an "XOR" (Exclusive OR) logic gate against the previous block before it is enciphered. The advantage to this mode is a significant obfuscation of the pre-encrypted data to prevent patterns from showing up in the encrypted result. An excellent explanation of the differences can be found here: http://en.wikipedia.org/wiki/Block_cipher_modes_of_operation. The downside to using CBC and similar modes of operation is increased processing time and implementation code complexity. **For simplicity and compatibility with the widest possible number of remote site implementations, our Authentication Schema uses Blowfish in ECB mode.** To avoid the problem of pattern based decryption, we have used the pre-encryption modifier described earlier in this document. Future versions of NCT Simple Signon will likely support CBC mode and other modes of encryption.

Padding, in encryption, is done to increase the length of the data to be encrypted to the next highest multiple of eight. For example, if a string to be encrypted is 29 bytes in length, three bytes are added to the string to make it 32 bytes long – a multiple of 8. If the string is already divisible by 8, no padding is done. There are two methods being used for padding in Blowfish implementations. The correct choice is to use a byte value equal to the number of padded characters. This is to say, if there are three padding bytes needed, each will be assigned the value of 3.

Cross-Platform Blowfish Resources

The best direct resource for finding program code, libraries, documentation, and products which use the Blowfish encryption algorithm is http://www.schneier.com/blowfish.html -- the author's web site.

Sample Packets

NCT Simple Signon includes the capability for administrators to create "sample" encrypted packets as needed to allow remote sites a test case during development and implementation. This will create a fully compliant packet of data including the username you have entered and a timestamp.

Blowfish Compatibility Test Sites

In addition to the test packet generator, NCT Simple Signon provides an easy method for testing a Blowfish algorithm for compatibility. The administrator of the NCT Simple Signon deployment can provide a URL which allows this testing. In addition, there are several other such resources on the web. At this time this document is being written, one such site is

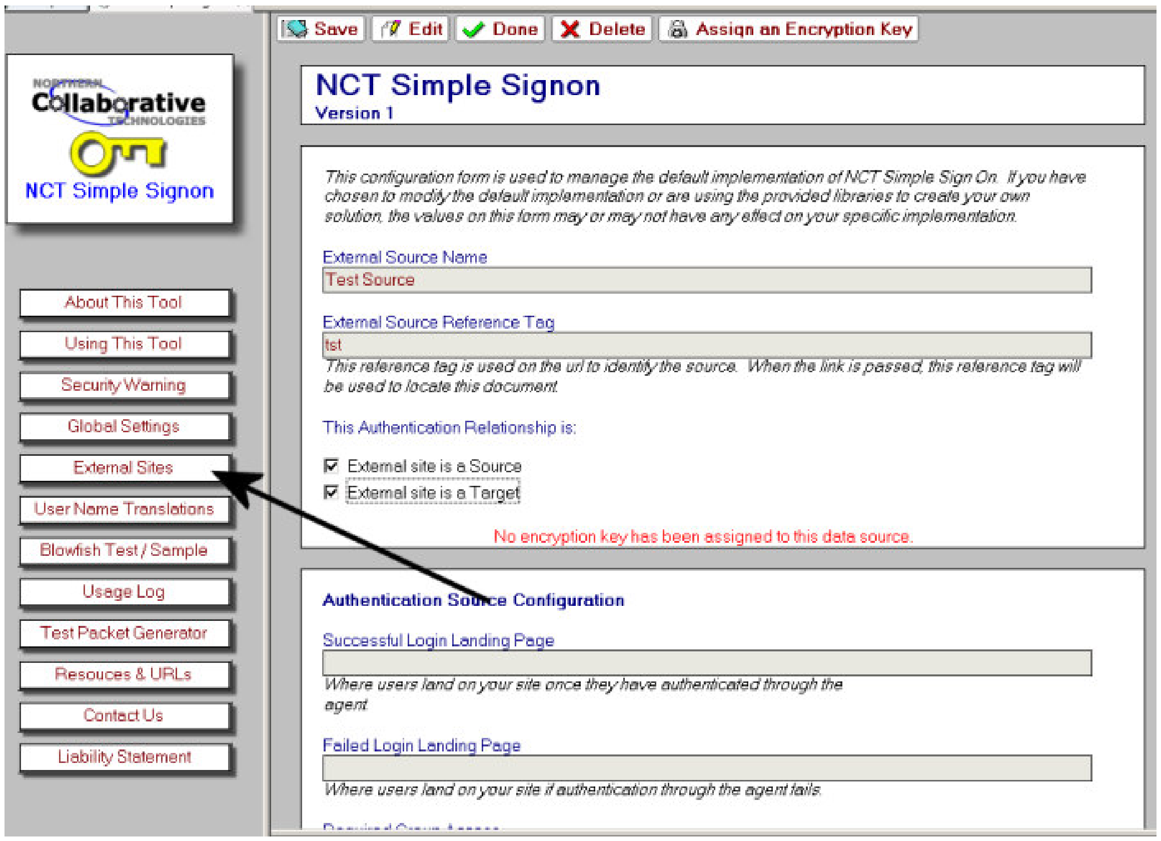
http://webnet77.com/cgi-bin/helpers/blowfish.pl

The site does have advertising sponsors, and as we don't own it or know the people who do, we cannot vouch for its safety. The best we can do is to tell you that "it seems to work for us".

CREATING A REMOTE SITE REFERENCE

To make implementation of NCT Simple Signon as easy as possible, our built in Authentication Schema is fully implemented for you. For each remote site you will connect with, create an "External Site" reference. This will allow you to define the location and parameters used for the site, as well as limit which users may use the site.

The first section allows you to define an arbitrary name for the remote site, and a site reference tag. The site reference tag will appear on the URL when passing users to the remote site, so it should be short and contain no spaces. This tag is used to match the site reference document with the inbound or outbound user as they use the tool. The check boxes are used to enable this site reference for passing users from your site to the remote site (target) or for accepting user inbound from the remote site (source) – or both.



NCT Simple Signon
Version 1

This configuration form is used to manage the default implementation of NCT Simple Sign On. If you have chosen to modify the default implementation or are using the provided libraries to create your own solution, the values on this form may or may not have any effect on your specific implementation.

External Source Name
Test Source

External Source Reference Tag
tst
This reference tag is used on the url to identify the source. When the link is passed, this reference tag will be used to locate this document.

This Authentication Relationship is:

☑ External site is a Source
☑ External site is a Target

No encryption key has been assigned to this data source.

Authentication Source Configuration

Successful Login Landing Page

Where users land on your site once they have authenticated through the agent

Failed Login Landing Page

Where users land on your site if authentication through the agent fails.

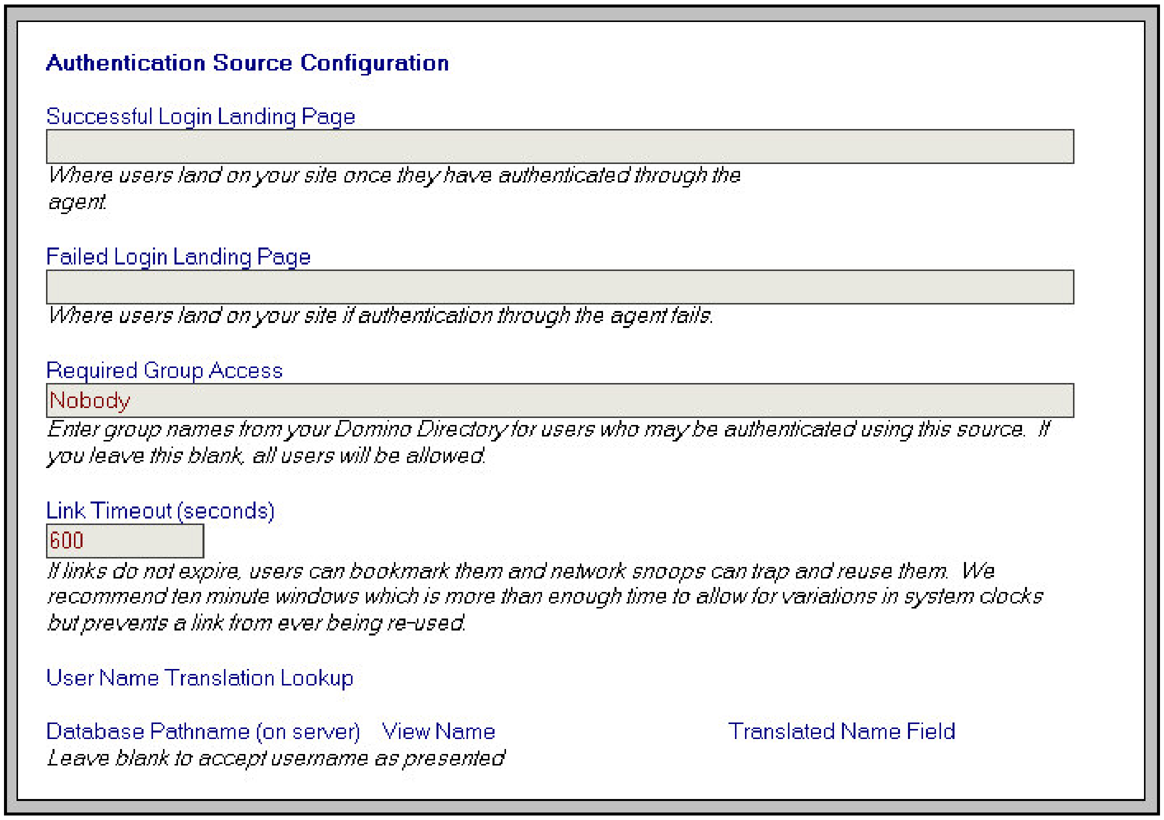
SOURCE SITE OPTIONS

When remote sites pass users to your site, they will always use the same URL. The URL they use is that of the receiving agent on your site. Once authenticated, the receiving agent will redirect the user to whatever page you specify below. This process is invisible to the user unless they are specifically looking for it.

When configuring your remote site as an authentication SOURCE, you will specify the landing page users should see when they pass through the process and are allowed access to your site, as well as an error page to redirect users who fail to authentication for any reason. In addition, you can (and should) specify one or more GROUPS from your Domino Directory which contain the names of users who are allowed to authenticate using this method. This is your best defense against misuse! By default, we list the group "Nobody". If this field is left blank, any user name may be authenticated using this method – including your name, or the names of your administrators.

The "TimeOut" field defines the length of time an encrypted packet of data is considered valid. We recommend starting at 10 minutes (600 seconds) as this is more than enough time to allow for difference in server clocks and network delays, yet is not enough of a window for the link with its packet included to be useful as a saved or forwarded link.

Finally, you have the option to translate usernames as they pass through your authentication agent by defining a database, view, and fieldname. Inbound users will be checked against the first column in your defined view. If a match is found, the text stored in the field name you have defined on that document will be used as the username for this inbound user. NCT Simple Signon includes a built in lookup mechanism, or you may use your own as needed.



Authentication Source Configuration

Successful Login Landing Page

Where users land on your site once they have authenticated through the agent.

Failed Login Landing Page

Where users land on your site if authentication through the agent fails.

Required Group Access

Nobody

Enter group names from your Domino Directory for users who may be authenticated using this source. If you leave this blank, all users will be allowed.

Link Timeout (seconds)

600

If links do not expire, users can bookmark them and network snoops can trap and reuse them. We recommend ten minute windows which is more than enough time to allow for variations in system clocks but prevents a link from ever being re-used.

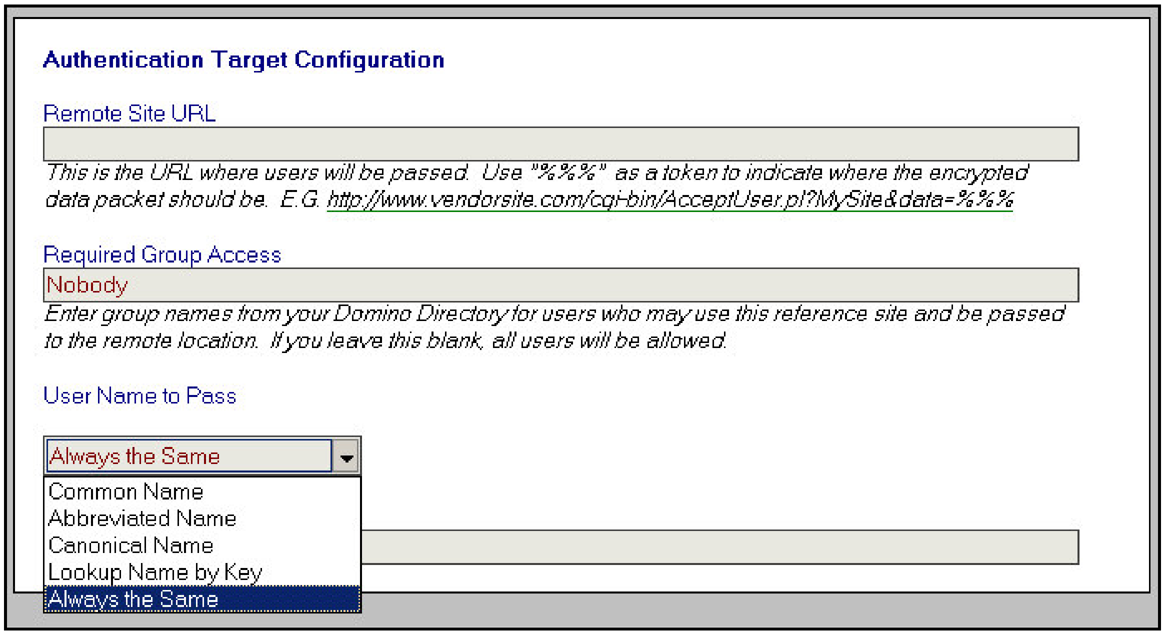
User Name Translation Lookup

Database Pathname (on server)   View Name   Translated Name Field

Leave blank to accept username as presented

TARGET SITE OPTIONS

Like the Authentication Source, the Target contains configuration options which include the URL on the remote site to pass the user to, and a group list which references the names of users who can use this remote site. In addition you may select which format of user name to pass to the site; or you may elect to always send the same user name – for sites with a single authentication, or use a translation lookup as in the case of the Source site.



Authentication Target Configuration

Remote Site URL

This is the URL where users will be passed. Use "%%%" as a token to indicate where the encrypted data packet should be. E.G. http://www.vendorsite.com/cgi-bin/AcceptUser.pl?MySite&data=%%%

Required Group Access

Nobody

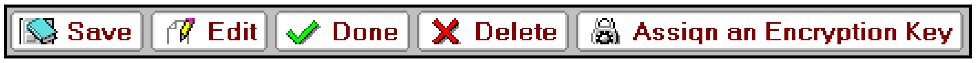
Enter group names from your Domino Directory for users who may use this reference site and be passed to the remote location. If you leave this blank, all users will be allowed.

User Name to Pass

Always the Same

SAFELY STORING AN ENCRYPTION KEY

An encryption schema is only as good as the safety of the agreed upon key. Our implementation of the built in Authentication Schema includes a way to safely store the shared key in a way that makes it inaccessible to other developers, administrators, or users on your system.



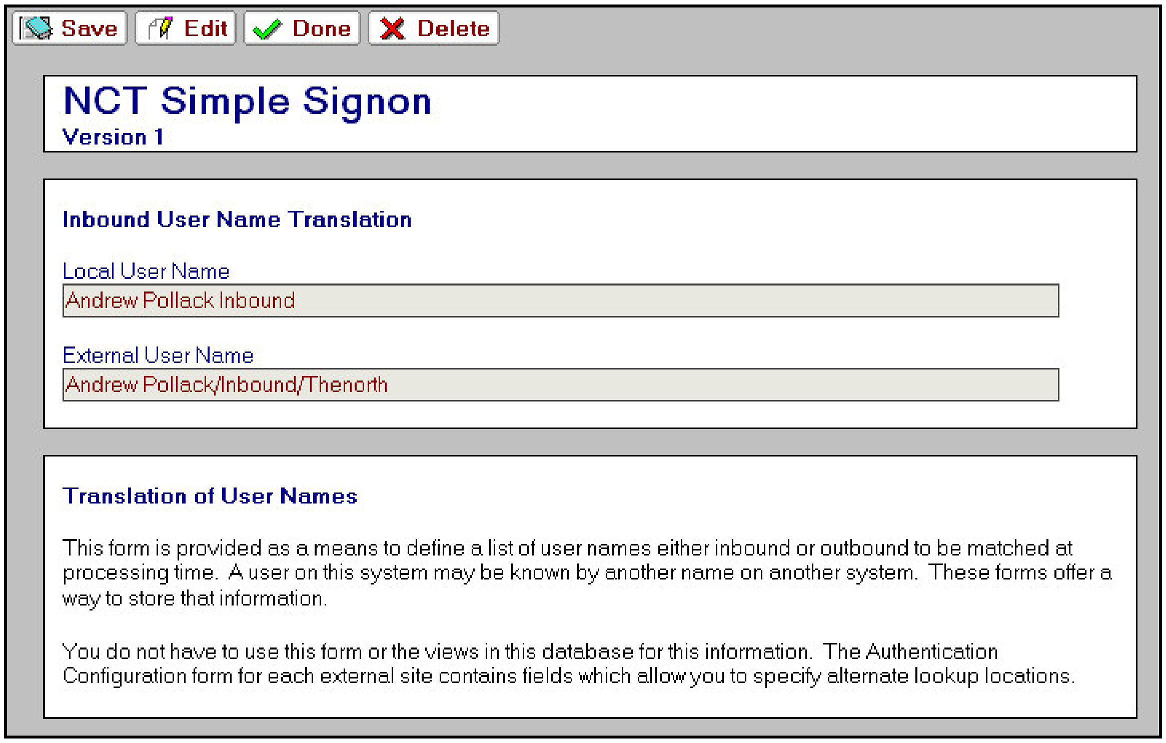
Selecting the option to assign an encryption key will cause the system to prompt you for a key to use, then store that key with the site reference document in a field which is itself encrypted using a secret key known only to NCT Simple Signon so that it can be used during the encryption process, but not directly accessed by a user or developer. Once the key is saved to the reference document, it cannot be viewed as plain text. Assigning a new key will overwrite any existing key.

In this interest of disclosure, it should be noted that the encryption used to protect this key is not entirely unique to your site, and could be used by a developer with access to the source code for NCT Simple Signon to decrypt your key. Of course, we protect this source code to the extent possible.

TRANSLATING USERNAMES

Using the built in Translation Lookup

For both inbound and outbound user name authentication, you have the option to translate usernames as they pass through your authentication agent by defining a database, view, and fieldname. Users will be checked against the first column in your defined view. If a match is found, the text stored in the field name you have defined on that document will be used as the username for this user. NCT Simple Signon includes a built in lookup mechanism, or you may use your own as needed.



NCT Simple Signon
Version 1

Inbound User Name Translation

Local User Name: Andrew Pollack Inbound

External User Name: Andrew Pollack/Inbound/Thenorth

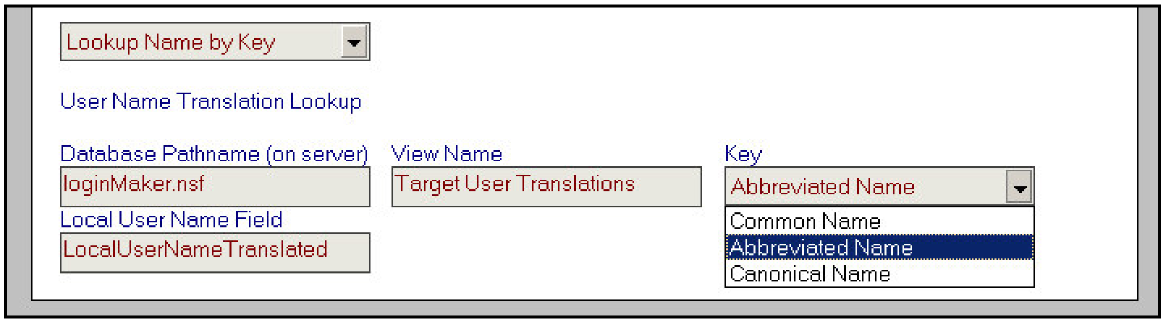
Translation of User Names

This form is provided as a means to define a list of user names either inbound or outbound to be matched at processing time. A user on this system may be known by another name on another system. These forms offer a way to store that information.

You do not have to use this form or the views in this database for this information. The Authentication Configuration form for each external site contains fields which allow you to specify alternate lookup locations.

Creating your own Translation Reference

By default, user name translation lookups are done by checking the built in translation documents. These are limited, however, and you may wish to define your own lookup database. If you choose the option to "Lookup Name By Key" you are presented with the optional fields to define the database, view, and key to use for the lookup as well as the field name which contains the actual value to use.



Lookup Name by Key

User Name Translation Lookup

Database Pathname (on server): loginMaker.nsf

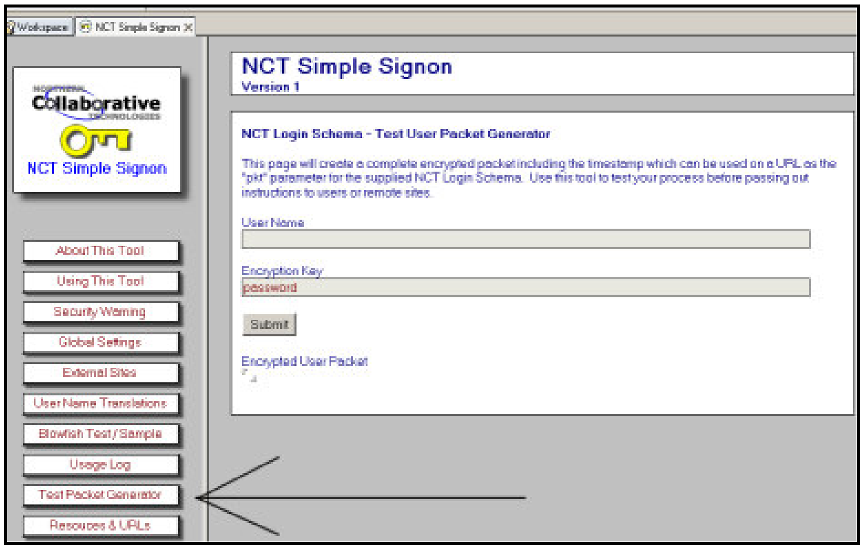
View Name: Target User Translations

Key: Abbreviated Name

Local User Name Field: LocalUserNameTranslated

PROVIDING SAMPLE PACKETS TO REMOTE SITES

NCT Simple Signon includes the capability for administrators to create "sample" encrypted packets as needed to allow remote sites a test case during development and implementation. Simply select the "Text Packet Generator" within the Application Database to create a fully compliant packet of data including the username you have entered and a timestamp.



NCT Simple Signon
Version 1

NCT Login Schema - Test User Packet Generator

This page will create a complete encrypted packet including the timestamp which can be used on a URL as the "pkt" parameter for the supplied NCT Login Schema. Use this tool to test your process before passing out instructions to users or remote sites.

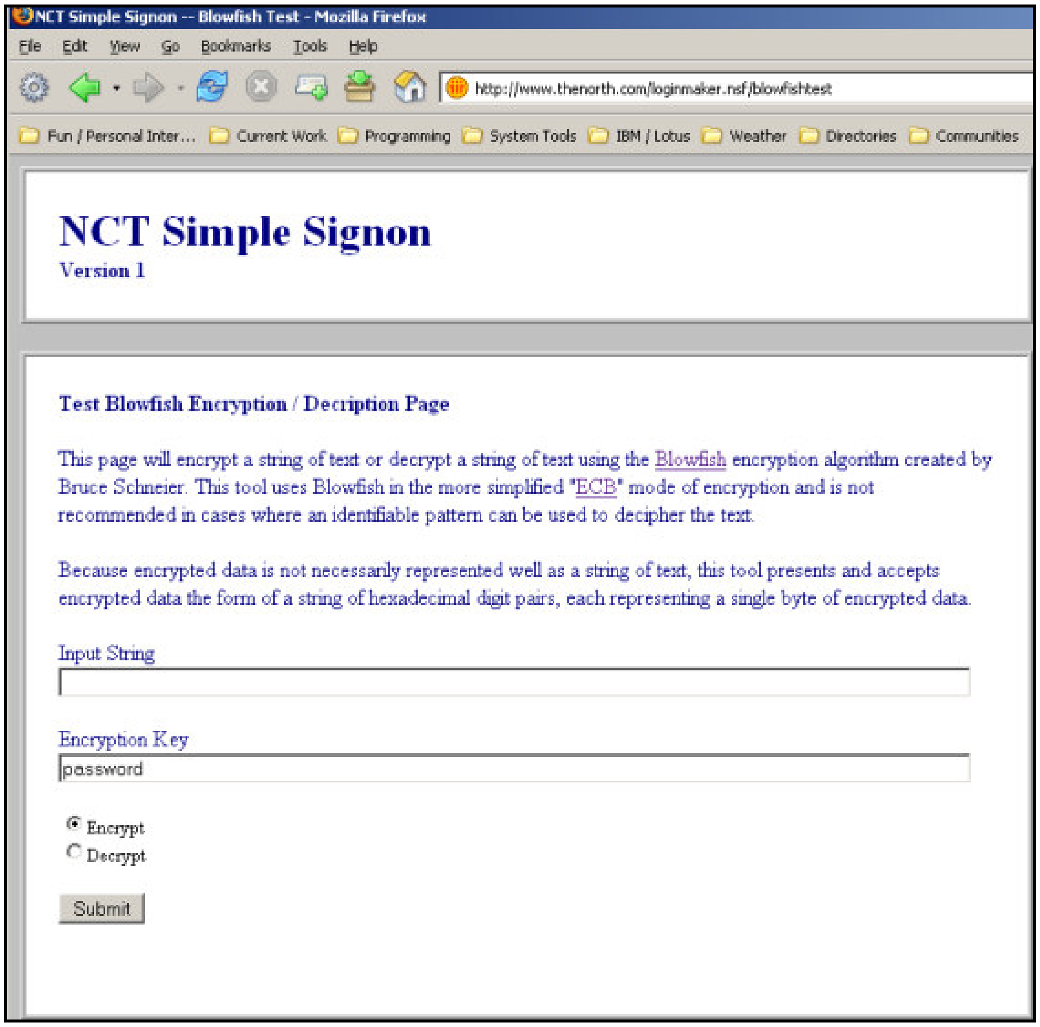
User Name

Encryption Key
password

Encrypted User Packet

ALLOWING REMOTE SITES TO TEST ENCRYPTION

In addition to the test packet generator, NCT Simple Signon provides an easy method for testing a Blowfish algorithm for compatibility. Select the “Blowfish Test / Sample” option in the Application Database to display a form which will encrypt or decrypt any string of text using the key you specify. This resource is also available using a browser, and can be given to remote sites or vendors to use in their testing. The URL will be the server and database pathname followed by ‘/blowfishtest’. For example: http://www.thenorth.com/loginmaker.nsf/blowfishtest.


NCT Simple Signon -- Blowfish Test - Mozilla Firefox
File Edit View Go Bookmarks Tools Help
http://www.thenorth.com/loginmaker.nsf/blowfishtest
Fun / Personal Inter... Current Work Programming System Tools IBM / Lotus Weather Directories Communities

NCT Simple Signon

Version 1

Test Blowfish Encryption / Decription Page

This page will encrypt a string of text or decrypt a string of text using the Blowfish encryption algorithm created by Bruce Schneier. This tool uses Blowfish in the more simplified "ECB" mode of encryption and is not recommended in cases where an identifiable pattern can be used to decipher the text.

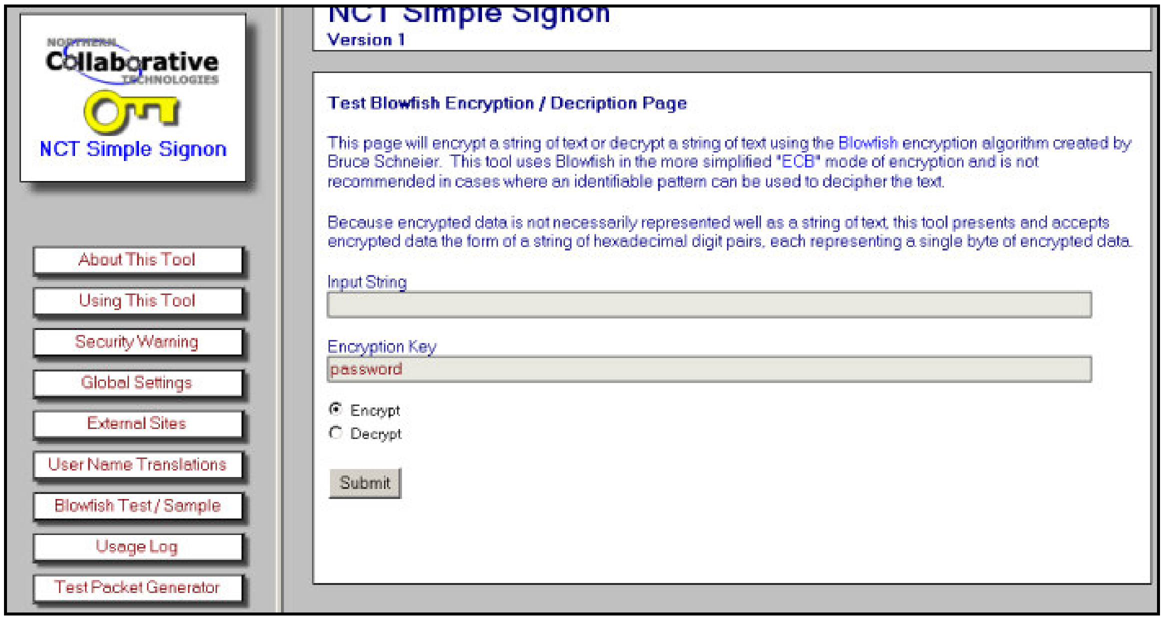
Because encrypted data is not necessarily represented well as a string of text, this tool presents and accepts encrypted data the form of a string of hexadecimal digit pairs, each representing a single byte of encrypted data.

Input String:

Encryption Key: password

[✓ Encrypt] Decrypt

Submit



Test Blowfish Encryption / Decription Page

This page will encrypt a string of text or decrypt a string of text using the Blowfish encryption algorithm created by Bruce Schneier. This tool uses Blowfish in the more simplified "ECB" mode of encryption and is not recommended in cases where an identifiable pattern can be used to decipher the text.

Because encrypted data is not necessarily represented well as a string of text, this tool presents and accepts encrypted data the form of a string of hexadecimal digit pairs, each representing a single byte of encrypted data.

Input String

Encryption Key
password

[✓ Encrypt]
Decrypt

Submit

USING THE API TO DEFINE YOUR OWN SCHEMA

If you've gotten all the way through this document and have not yet been dissuaded from creating your own authentication mechanism, this section defines what you'll need at a minimum.

THE BARE MINIMUM

At the barest minimum, to use the provided tool you need the following:

1. The DLL file provided with the installation must be present in your Domino server's program directory.

2. The License Key file provided with the installation must be present in your Domino server's program directory.

3. You must include the Script Library "NCT_SingleSignOn_MasterLibrary" from the Application Database in your application, and reference it in your code.

If you have done these things, and your server is properly configured for single sign on, you may use the call:

Public Function nct_createLtpaToken(Username As NotesName, LTPAorgName As String, LTPAconfigName As String) As String

The following are the parameter values:

Username - A string variable containing the name of the user to create the LTPA token for. This should be a hierarchical or canonical format name if possible.

LTPAorgName - A string variable which must match the organization specified in your Web SSO document, within the Domino Directory.

LTPAconfigName - A string variable which must match the LTPA Token Name you have defined in the Web SSO document, within the Domino Directory.

Returns:

The function will return an empty string if there is an error, or a string of ascii text which is an LTPA token. To assign this token to a user, set a temporary cookie in the user's browser with the name "LtpaToken" and assign this string as the value.

Notes:

1. If your license key is not valid, this string may be empty or may contain invalid data. A valid LTPA token will be in base64 encoding. It should be left that way, as it is what the Domino server will be looking for.

2. You must sign any agent which uses this tool, using an ID that has the authority to run "Unrestricted LotusScript Agents"

3. **This is a potentially damaging action. You are causing a specific CAPI call to be made. Although we have taken steps to prevent it, passing incorrect parameters to this function could result in a server crash. We recommend this be done in a test environment first.**

ADDITIONAL RESOURCES

In addition to the bare minimum requirements, the following resources are available for you to create your own authentication process using our tool:

First, view the source code. The code which handles the cookie assignment and redirection is readily visible. Feel free to use or alter it – but of course we have to warn you, you're on your own.

These additional public function calls are included in the master library, and may be of use.

General Use

Public Function nct_getparameter(source As String , key As String) As String

When passed a string – typically a URL – and a key, will return a string which starts after the end of the first occurrence of the key in the source string and ends at the last character before the end of the string or an "&" is encountered. This is used to extract parameters from a URL.

Public Function nct_isUserInOneOfTheseGroups(username As String, groupArray As Variant, server As String) As Boolean

This function accepts a user name as the first string, and an array of strings which contain a set of groups to check. This will recursively check to see if the user name appears in any of the groups, or group which is nested within the groups to a nesting depth of six levels. When the user name supplied is hierarchical, all valid wildcard versions and representations of the name (e.g. Abbreviated, Canonical, Common, etc.) are checked as well.

Blowfish Encryption

Public Function nct_blowfish_unpack(Byval text As String, Byval pwd As String) As String

This function accepts a string of text which consists of an even number of characters, all of which are valid hexadecimal digits and attempts to decrypt the string using the Blowfish ECB algorithm using the supplied password as the key.

Public Function nct_blowfish_pack(Byval text As String, Byval pwd As String)As String

This function accepts a string of text and an encryption key string, and returns the value encrypted using the Blowfish ECB encryption algorithm represented as an even number of characters, all of which are valid hexadecimal digits.

Built In Authentication Schema Functions

The following function calls are useful only if you are using the built in Authentication Schema reference documents.

Public Function nct_translateInboundName(username As String, refdoc As notesdocument) As String

This function accepts a user name and an Authentication Schema reference document. It will perform any needed user name translations according to the configuration of that reference document for inbound user names.

Public Function nct_translateOutboundName(username As String, refdoc As notesdocument) As String

This function accepts a user name and an Authentication Schema reference document. It will perform any needed user name translations according to the configuration of that reference document for outbound user names.

Public Function nct_schema_makepkt(payloadstring As String, refdoc As NotesDocument) As String

This function accepts a "payload string" (typically a user name) and an Authentication Schema reference document. It will return a valid encrypted packet with the proper timestamp encrypted using the stored encryption key from the External Site Reference Document. This function should be called AFTER the name translation is complete. If no encryption key has been stored, this function will generate unreadable results.

Public Function NCT_Schema_UnpackPkt(Byval pkt As String, refdoc As notesdocument) As Boolean

This function accepts an encrypted packet, and will attempt to decrypt the packet using the stored encryption key from the External Site Reference Document. If no encryption key has been stored, this function will fail. Results will only be returned if the packet can be decrypted using the supplied key, the current license key is valid, and the timestamp is valid for use within the time specified on the external site reference document. In the case of an out of date packet, no data will be returned.