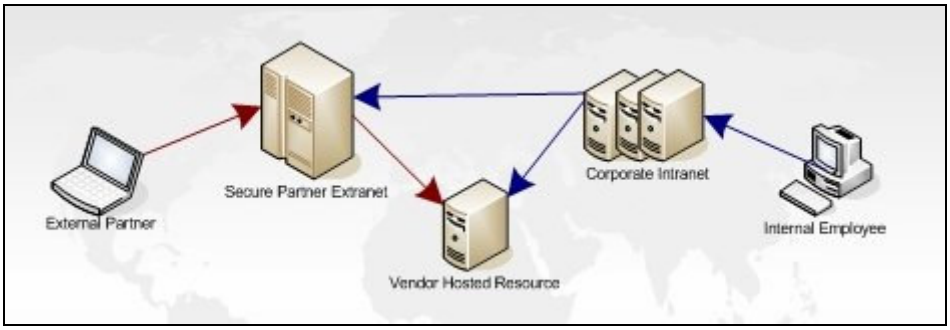
NCT SIMPLE SIGNON

INTEGRATION WITHOUT AGGRAVATION



Remote Site Authentication Schema

This document is designed to be given to remote sites which will be integrating with an IBM Lotus Domino based web site which is using NCT Simple Signon with its built in Authentication Schema to allow transfer of users to and from other sites without presenting a request for credentials. It contains sufficient details on the process of integration from a **vendor neutral** perspective.

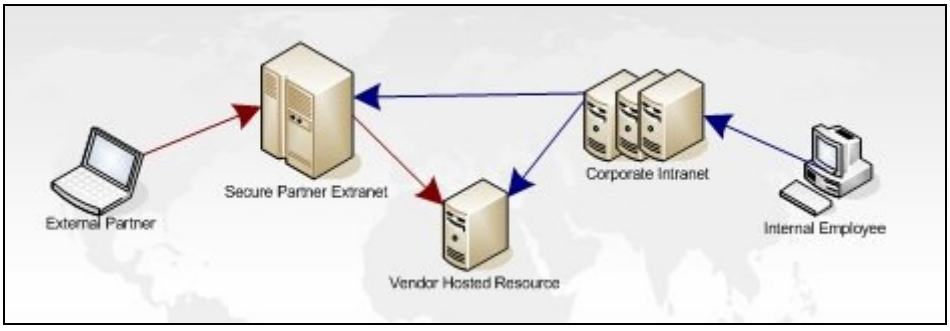
NORTHERN Collaborative TECHNOLOGIES

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.

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.

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. Here's the short list we came up with:

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.

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.

PROCESS FLOW

Using this schema, users who are authenticated on one web site click a link which is a reference to another. The link itself, however, does not go directly to the remote site. It goes to a script or program on the current site which generates an encrypted packet of data containing the

user's name and a timestamp. This packet of data is then used as part of a URL reference to the remote site. Once the customized URL is created, the user is redirected to that URL. On the remote site, another script or program reads the encrypted packet of data, validates the timestamp, assigns a local authentication token to the user and redirects the user to the correct landing page. It may sound like a lot of steps, but to the user, it is transparent other than a brief delay. Most users will not even notice the redirection. For those who do, it will not be unusually long or intrusive. Users simply click a link, wait a moment or two, and see the correct page on the remote site.

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 (which is the implementation in the IBM Lotus Domino side of the transaction) 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